RECONSTRUCCIÓN DE DECISIONES PLATAFORMA FINANCIERA MIFOS X CON TÉCNICAS DE ADMINISTRACIÓN DE CONOCIMIENTO

IVAN ANDRES JIMENEZ ROA
LUIS FERNANDO ROJAS ORTIZ

UNIVERSIDAD PILOTO DE COLOMBIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ
2017

RECONSTRUCCIÓN DE DECISIONES PLATAFORMA FINANCIERA MIFOS X CON TÉCNICAS DE ADMINISTRACIÓN DE CONOCIMIENTO

IVAN ANDRES JIMENEZ ROA
Cód. 1110288.
LUIS FERNANDO ROJAS ORTIZ
Cód. 1010314.

Proyecto de grado para optar al título de ingeniero de sistemas

Director
Gilberto Pedraza García
Ingeniero de Sistemas y Computación

UNIVERSIDAD PILOTO DE COLOMBIA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ
2017

Nota de aceptación:

_____
_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Bogotá D.C. 12 de Junio del 2017

Éste trabajo está dedica a nuestros padres y familiares quienes brindaron todo su apoyo incluso en los momentos más difíciles y que jamás dejaron de creen en nosotros. También dedicamos esté trabajo a personas cercanas tanto amigos como docentes quienes nos demostraron a lo largo del camino que vale la pena luchar por nuestros objetivos.

AGRADECIMIENTOS

Queremos agradecer al Ing. Gilberto Pedraza García, docente del programa ingeniería de sistemas de la Universidad Piloto de Colombia. Sin su constante apoyo este trabajo no hubiese sido posible.

Igualmente queremos agradecer al sociólogo Ignacio Hernández Molina, docente del programa ingeniería de sistemas de la Universidad Piloto de Colombia, por su apoyo, sabios consejos e ideas que ayudaron a la inspiración de éste trabajo.

Finalmente extendemos nuestra gratitud a personas que conocimos en el transcurso de nuestra carrera quienes compartieron sus conocimiento y experiencias formando un vínculo de amistad sólido, en especial al ingeniero Leonardo Escobar Archila, profe gracias por compartir sus conocimientos con nosotros.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

GLOSARIO

AngularJS: es un *framework* de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

Clase: se llama clase a la declaración o abstracción de un objeto cuando se programa según el paradigma de orientación a objetos.

Diagrama: Un diagrama es un gráfico que presenta en forma esquematizada información relativa e inherente a algún tipo de ámbito, como ser la política o la economía de alguna nación o empresa y que aparecerá representada numéricamente y en formato tabulado.

Eclipse: es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

IDE: entorno de desarrollo interactivo, en inglés *Integrated Development Environment* (**IDE**), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

Java: es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

Mapeo: técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

Modelo: representación de procesos, modelos o sistemas que conforman un conglomerado mayor o supra-sistema, que pretende el análisis de interacción de ellos, a fin de mantener una relación flexible que les permita cumplir su función particular y coadyuvar para cumplir la función del supra-sistema.

Módulo: es una porción de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará, comúnmente, una de dichas tareas (o varias, en algún caso).

Open Source: También conocido como código abierto es el *software* desarrollado y distribuido libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.

UML: el lenguaje modelo unificado (*Unified Modeling Lenguage*) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Repositorio: un repositorio, depósito o archivo es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

Requerimiento: En la ingeniería de sistemas, un requisito es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Se usa en un sentido formal en la ingeniería de sistemas, ingeniería de software e ingeniería de requisitos.

Versión: el versionado de software es el proceso de asignación de un nombre, código o número único, a un software para indicar su nivel de desarrollo.

RESUMEN

MIFOS X es una plataforma desarrollada para el sector financiero que actúa como un core bancario capaz de soportar múltiples servicios, entre los más destacables se encuentra el manejo a nivel corporativo de cuentas bancarias, administración de clientes, manejo general de créditos entre otros servicios, esta plataforma fue desarrollada bajo licencia de código abierto, con la finalidad de que personas u organizaciones interesadas en contar con los servicios de MIFOS X puedan adecuar sus propias necesidades y así incluir dicha plataforma dentro de sus organizaciones.

Por medio sitios oficiales que recopila una cantidad significativa de información tanto de la herramienta como de la iniciativa MIFOS responsable del inicio del desarrollo de la herramienta, generando de esta manera una ayuda tanto teórica como técnica para usuarios que no tienen el mínimo conocimiento de la herramienta, este tipo de ayuda es de utilidad significativa pero surge la siguiente pregunta: ¿Qué sucedería si esa información no se encuentra organizada a simple vista de los usuarios sino que por el contrario esta se encuentra dispersa en diversos sitios en donde la búsqueda por parte de los usuarios se vuelve tediosa y difícil de encontrar?. La respuesta abierta es que dicha información no sirve como ayuda debido a su dificultad de entendimiento y ubicación, por otra parte, es fundamental que los usuarios conozcan al detalle cuales fueron las decisiones que llevaron a los creadores de MIFOS X a desarrollar cada uno de los módulos de esta herramienta.

Mediante técnicas de ciclo de vida del conocimiento e ingeniería inversa, se administrará un estructura de información cuyo objetivo sea proporcionar al usuario la información de tal manera que este puede asimilarla de una mejor manera, así como de reconstruir parte de la aplicación tomando como base el código de la aplicación y generar un mayor conocimiento acerca de los diferentes módulos implementados en MIFOS X por medio del diseño de una Wiki semántica enfocada únicamente a MIFOS X detallando lo mencionado anteriormente.

Palabras clave: MIFOS, MIFOS X, *core banking,* wiki semántica, administración de conocimiento, ciclo de vida del conocimiento, ingeniería inversa.

INTRODUCCIÓN

Las técnicas de ciclo de vida del conocimiento, son herramientas que permiten a una organización mantener un control y gestión de los procesos de desarrollo y diseño de sus productos, para así poder disponer de esa información para futuras necesidades que se presenten, no obstante este tipo de gestión resulta ser muy costosa y poco beneficiosa para algunas compañías debido a que los profesionales en cada etapa de desarrollo de un producto en este caso de Software, toma como propio su conocimiento y experiencia que adquirió resolviendo una necesidad, haciendo de dicha experiencia y conocimiento adquirido que muchas veces no se valora un componente esencial, el cual se llega a perder o simplemente queda en el olvido, obligando a las compañías de diseño de software replicar procesos que en muchas de las ocasiones se habían desarrollado en anteriores proyectos.

MIFOS X por otro lado es una plataforma de servicios financieros con la suficiente adaptabilidad para implementar futuras mejoras, no obstante entender como fue desarrollado un productor de software, es una tarea bastante dispendiosa, la cual requiere de tiempo el cual se debe abordar múltiples técnicas entre las más comunes se encuentran la ingeniería inversa, ciclo de vida del conocimiento, entre otras, para lograr incorporar o mejorar módulos y hacer de MIFOS X una importante herramienta de gestión y administración de servicios financieros.

Por esa razón para construir ese conocimiento, se decide enfocar la investigación realizada en las comunidades involucradas en desarrollo e implementación de MIFOS X, con el fin de modelar dicha información a partir de ese conocimiento adquirido, y generar un conocimiento entendible a usuarios que desean conocer más acerca de la herramienta y realizar sus respectivos desarrollos adaptando necesidades propias en la plataforma MIFOS X.

1. GENERALIDADES

1.1 DESCRIPCIÓN DEL PROBLEMA

Actualmente existen muchas herramientas de software de código abierto que permiten hacer extensiones o adecuaciones a necesidades específicas. Aunque existe documentación de las herramientas esta es muy general y da cuenta de los modelos y diagramas finales de arquitectura. En algunos casos no se encuentra documentado el diseño y solamente se cuenta con el código ejecutable.

MIFOS X es una plataforma de tecnología abierta para la inclusión financiera que proporciona la funcionalidad básica necesaria para ofrecer servicios financieros a los 2 mil millones de pobres y no bancarizados.

MIFOS X es una distribución estándar que contiene la plataforma, cliente web y cliente móvil. Se puede implementar en cualquier entorno: cloudoon-premise, offline, móvil o PC; Es suficientemente extensible para soportar cualquier tipo de organización o canal de distribución y lo suficientemente flexible como para soportar cualquier producto, servicio o metodología. Para cualquier organización, grande o pequeña, proporcionará la gestión de datos del cliente, la gestión de préstamos y cartera de ahorros, la contabilidad en tiempo real integrada y la información social y financiera deben traer servicios financieros digitales en un mundo conectado moderno.

El motor flexible de creación de productos le permite crear, configurar y lanzar al instante nuevos productos de crédito, depositar y compartir productos a través de cualquiera de sus sucursales. Se puede responder a la demanda del cliente en cualquier momento - una vez que realice el cambio en **MIFOS X**, los nuevos productos están disponibles para cualquier cliente a través de cualquiera de sus sucursales. También puede configurar la combinación de productos para garantizar que los clientes sólo pueden sacar una cantidad responsable de préstamos al mismo tiempo.

La API de **MIFOS X** admite:

- Préstamos individuales
- Préstamos del grupo de responsabilidad conjunta de Grameen
- Préstamos de varios tramos
- Préstamos a Plazo Variable para la Agricultura
- Préstamos de tipo de interés flotante
- Instrucciones permanentes para pagos automatizados
- Préstamos cooperativos

- Todas estas metodologías están incluidas en el paquete de productos **MIFOS X**.

Para préstamos de grupo, tenemos plena funcionalidad para configurar y administrar grupos, incluyendo información de clientes, grupos y centros, y jerarquía de grupos y centros.

El ahorro ya está emergiendo como uno de los servicios más probados en el kit de inclusión financiera. **MIFOS X** apoya a su organización ya que se mueve más allá del ahorro básico de la libreta de ahorros con el apoyo a depósitos fijos y recurrentes, cuentas corrientes, instrucciones permanentes, transferencias de cuentas para que su cliente pueda beneficiarse de una mezcla fluida de productos de depósito y crédito para almacenar los fondos. Resistir los choques y aumentar los ingresos necesarios para hacer mayores compras e inversiones.

- Ahorros en la libreta
- Depósitos a plazo fijos
- Depósitos recurrentes
- Cuentas actuales
- Acciones Cuentas y Dividendos

MIFOS X soporta el cálculo del saldo de intereses sobre el saldo diario o promedio con la posibilidad de especificar el período de composición mensual o diaria y el período de contabilización mensual, trimestral o anual.

Apoyo a las tarifas planas o porcentuales que se pueden aplicar y cobrar en una fecha de vencimiento específica, al activarse, al retirarse o sobregiro como honorarios mensuales o anuales. Apoyo para un saldo mínimo de apertura como un período de bloqueo que se puede definir en días, semanas, meses y años.

La ingeniería inversa es el proceso de extracción de conocimiento o información de diseño de cualquier cosa hecha por el hombre y volver a producirlo o volver a producir cualquier cosa sobre la base de la información extraída. En ingeniería de software la ingeniería inversa es “el proceso de analizar un sistema de software para crear representaciones del sistema en un nivel más alto de abstracción”[1]. Algunos Autores la consideran como "retroceder a través del ciclo de desarrollo", de forma que la salida de la etapa

[1] RAMOS-ACOSTA. Diego Alonso, Proceso de ingeniería inversa. [abril 2013]. Colombia. <https://www.acofipapers.org/index.php/acofipapers/2013/paper/viewFile/380/189> [Citado marzo 23 de 2017].

de implementación (código objeto) se invierte hacía el diseño y análisis de requerimientos.

1.2 PREGUNTA INVESTIGATIVA

¿Por qué aplicar procesos de recuperación de información como lo son ingeniería inversa y ciclo de vida del conocimiento a la plataforma MIFOS X?
MIFOS X es un software de implementación financiera gratuito, la ventaja principal de esta plataforma consiste en ser una tecnología de código abierto (*open source*), cuyo objetivo principal consiste en apoyar los esfuerzos de inclusión financiera a nivel mundial y que provee la funcionalidad básica requerida para la entrega de los servicios financieros a más de 2 billones de personas alrededor del mundo, especialmente en países del tercer mundo en África, Asia y Latinoamérica.

La versión que actualmente se puede encontrar de la plataforma **MIFOS X** cuenta con una funcionalidad básica, por lo cual esta debe ser modificada según los parámetros y necesidades que se soliciten, es allí donde se inicia la labor de ingeniería la cual consta en realizar un estudio completo de la herramienta desde sus origines lo cual permita entender en su mayor parte el código fuente, brindando así el conocimiento y la facultad de realizar cambios en la herramienta.

Al día de hoy la versión base de **MIFOS X** ha pasado por varias modificaciones, lo cual supone inconvenientes ya que dichas modificaciones que se realizaron versión tras versión no se encuentran documentadas, por lo que dificultad la compresión del código de la herramienta, adicional es bastante limitada la información que se encuentra de dicha herramienta, debido a esto la herramienta presenta un nivel alto de dificultad al momento de analizar su código base, tanto es así que incluso para instalar la herramienta en modo de desarrollador es un proceso bastante complejo y que la actual información que se puede encontrar de este proceso es confuso.

Dicho lo anterior la herramienta **MIFOS X** necesita de procesos como ingeniería inversa y/o ciclo de vida del conocimiento según sea necesario, lo cual a futuro permitirá la correcta modificación de esta misma, así reduciendo tiempos de investigación y pasar a realizar las modificaciones directamente según se indique, esta herramienta será una iniciativa que ayudará principalmente a compañías financieras medianas y pequeñas en su desarrollo empresarial.

1.3 JUSTIFICACIÓN

En la actualidad existe un número significativo de compañías financieras medianas y pequeñas que necesitan disponer de los servicios de un core

bancario lo cual implica en mucho casos pagar el licenciamiento de estos mismos, en muchos casos los core bancarios licenciados cuentan con procesos estandarizados esto supone un problema debido a que no se puede acomodar completamente a las necesidades de una compañía determinada y el realizar cambios a los core bancarios licenciados supone en la mayoría de las ocasiones costos elevados.

Ejemplo de esto es el sistema de coreBacking Temenos T24 el cual es uncore bancario que es bastante común en empresas financieras medianas. Desarrollado en Londres en el año de 1993, el cual presta servicios de desarrollo de software para el sector financiero y de banca, su licenciamiento es costoso y puede no llegar a cubrir las necesidades completas de una empresa financiera, esto indica que migrar todas las operaciones financieras por parte de una compañía financiera a una solución de software libre MIFOS X lo que implica ahorro y funcionalidad controlable.

En concreto MIFOS es una iniciativa de la Fundación Grameen y a la vez un software financiero producido para la industria de las micro-finanzas. El software proporciona funcionalidad clave para instituciones de micro-finanzas.

Por este motivo, el tener acceso al código fuente de su core bancario y específicamente comprender tanto su parte técnica como teórica, representa para a compañía una gran ventaja ya que se libera de la dependencia tecnológica de terceros y queda en total autonomía para la modificación del mismo de manera que la compañía podrá ajustarse de manera ágil a nuevos requerimientos y normativas legales en el futuro, siempre teniendo a la mano procesos adecuados de ingeniería inversa y ciclo de vida del conocimiento lo cual facilitará el desarrollo de la herramienta de manera correcta y óptima.

1.4 ALCANCE

Compañías financieras que deseen implementar la plataforma de código abierto MIFOS X y/o para aquellos estudiantes que deseen seguir con esta temática y brindar un producto óptimo y desarrollado correctamente.

El presente proyecto está dirigido a la realización de procesos de ingeniería inversa y ciclo de vida del conocimiento de la plataforma MIFOS X el cual presenta una documentación detallada sobre la investigación que se realizará de la herramienta, lo cual se acompañará con utilización de diagramas, manuales y otras ayudas que permitan la comprensión adecuada de la herramienta y facilite su futuro desarrollo.

Por lo que se considera como alcance del proyecto los siguientes puntos:

- Entendimiento de la plataforma MIFOS X.
- Organización de información recopilada.

- Proceso de ingeniería inversa aplicado a la plataforma MIFOS X.
- Aplicación de técnicas de ciclo de vida del conocimiento a la plataforma MIFOS X.
- Representación de las técnicas aplicadas a la herramienta.

1.5 LÍMITES

En los límites del proyecto se realizará una investigación detallada junto con procesos de ingeniería inversa y administración de información de la herramienta MIFOS X, ya que enfrentarse directamente al desarrollo de la herramienta sin conocimiento previo puede generar confusión en cuanto al código presentar un resultado final con malas prácticas de desarrollo por lo que se desea evitar esto realizando una previa investigación, ya que la herramienta es la base para aportar grandes avances al sector financiero.

Es de vital importancia mencionar que al ser un tema meramente investigativo no se presentará producto desarrollado más en concreto modificación del código ya que esto implica un incremento en el tiempo y alcance con lo cual no se cuenta, por otro lado, se presentará todos los resultados recopilados de la investigación por medio de una wiki semántica ya que es el mejor medio presentar este tipo de información, ya que a futuro usuarios externos interesados en el tema pueden generar aportes en esta.

MIFOS X es una plataforma de código abierto para compañías financieras la cual está desarrollada en plataforma Java nativo para su BACK-END, y otra combinación de lenguajes teniendo como principal frame AngularJS para su FRONT-END.

1.6 OBJETIVOS

1.6.1 Objetivos generales. Establecer una estrategia de recuperación de las principales decisiones de diseño de una plataforma de código abierto para hacer comprensible el diseño y de forma que permita tomar nuevas decisiones para modificar, actualizar, evolucionar y mantener el producto.

En particular se pretende implementar metodologías de ingeniería inversa y ciclo de vida del conocimiento con el fin de realizar una investigación a fondo de la herramienta cuyo objetivo es asociar los métodos, arquitectura y documentación que se llevó a cabo para el desarrollo del core Bancario Conocido como MIFOS X**.**

1.6.2 Objetivos específicos. Determinar mediante un catálogo de información, cada una de las etapas de desarrollo que se llevaron a cabo para implementar la plataforma MIFOS X.

Probar múltiples herramientas que desarrollen técnicas de ingeniería inversa, para validar cuál de todas dependiendo de sus características y propiedades serán la más adecuada para realizar un proceso de reingeniería en una aplicación ya desarrollada y de la cual se desconocen muchos de sus procesos.

Aplicar técnicas de ciclo de vida del conocimiento con el fin de ubicar anotaciones referentes al proyecto que hayan generado impacto significativo entre los usuarios de la comunidad desarrolladora de la herramienta llevando a múltiples decisiones tomadas para el desarrollo de esta misma.

Tras la realización de técnicas de reingeniería de la herramienta MIFOS X y recopilando toda la información de tal manera que el usuario pueda tener a la mano esta ayuda realizar pruebas con el fin de que una vez se haya pasado por el análisis de la información proporcionada la modificación de la herramienta MIFOS X se facilite notablemente.

2. MARCO TEÓRICO

2.1 MARCO REFERENCIAL

2.1.1 Core Banking. A través del tiempo y con los avances tecnológicos que han ido surgiendo las diferentes áreas profesionales han decidido implementar esta misma en sus diferentes áreas de investigación, trabajo, procesos, etc. Y el área financiera no es la excepción, por lo cual se decide realizar implementación del denominado *"core bancario"*, cuya función radica en una plataforma donde se combinan la tecnología de la comunicación y la tecnología de la información para satisfacer las necesidades del negocio y de la industria bancaria también se puede definir como el negocio desarrollado por una institución bancaria con sus clientes minoristas y pequeñas empresas. Muchos bancos tratan a los clientes minoristas como a sus clientes de "Core bancario", y tienen una línea de negocios separada para gestionar las pequeñas empresas. Las grandes empresas son administradas a través de la División de Banca Corporativa de la institución. "Core bancario", básicamente, se refiere a las operaciones de depósito y de préstamos de dinero. Normalmente las funciones de Core bancario incluyen cuentas de depósito, préstamos, hipotecas y pagos. Los bancos hacen estos servicios disponibles al cliente a través de múltiples canales como cajero automático (ATM), puntos de venta (POS), banca por Internet, entre otros.

Con el surgimiento de varias empresas pequeñas dedicadas al financiamiento se decide desarrollar varios proyectos de software libre para esta temática, con el fin de que las compañías minoristas puedan hacer uso de esta tecnología y adecuarlo a sus necesidades lo que es denominado "Open Source" cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de Autores, son publicados bajo una licencia de software compatible con la Open SourceDefinition o forman parte del dominio público. Esto permite a los usuarios utilizar, cambiar, mejorar el software y redistribuirlo, ya sea en su forma modificada o en su forma original. Frecuentemente se desarrolla de manera colaborativa y los resultados se publican en internet. El software es el mejor ejemplo del desarrollo del código abierto y se compara con el llamado "contenido generado por los usuarios". La expresión *software de código abierto* surgió a partir de una campaña de mercadotecnia para el software libre. Un informe del StandishGroup afirma que la incorporación de los modelos de software de código abierto ha resultado en ahorros de aproximadamente 60 mil millones de dólares por año a los consumidores.[2]

[2] BANKING OFFERING. accenture consulting. [Junio de 2014]. United Kingdom. < https://www.accenture.com/gb-en/core-banking-services> [Citado febrero 12 de 2017].

2.1.2 Iniciativa MIFOS. Con el tema de open source que abarca en general cual área profesional, toma cartas en el asunto y como ayuda para el área de financiamiento la iniciativa *"MIFOS"*, impulsada por la comunidad que se está posicionando las instituciones financieras para convertirse en proveedores modernos y conectados digitalmente de servicios financieros a los pobres. Abierto a todos y apoyado por una comunidad a nivel mundial, que permiten a los nuevos modelos de negocio que crear y distribuir valor en todo el sector. A través de una plataforma común apoyada por una comunidad de colaboración, se obtiene un mayor rendimiento para la MIFOS para ayuda de todo el sector de la inclusión financiera.

2.1.3 Plataforma MIFOS X. Las herramientas para que esto ocurra son un conjunto de tecnologías que si se es un usuario que busca el software libre, la herramienta MIFOS X es la adecuada, o si se es un desarrollador que buscan construir su propia solución para la banca móvil o servicios financieros digitales la herramienta también es bastante viable. La herramienta es completamente abierta, la plataforma proporciona una funcionalidad inclusión financiera central, totalmente expuesta a través de las API REST, y lo mejor es que su funcionamiento es fácil de entender.

En cuanto a la plataforma MIFOS X se debe tener en cuenta los siguientes tres aspectos de vital importancia:

- **Plataforma**. MIFOS X, una plataforma moderna con una arquitectura orientada al servicio modular que proporciona toda la funcionalidad común necesaria para ofrecer servicios financieros digitales. Un back-end REST que habla a un a una base de datos MySQL y se crea en el / la primavera / pila APP Jersey cuyo frontal está completamente desacoplado para que pueda elegir la capa de presentación de su elección.

- **App**. EL producto principal del aprovechamiento de la plataforma MIFOS X, construido en AngularJS, un contenido en Javascript cuya construcción de aplicaciones se simplifica en sola página rápida. Proporciona soporte para los modelos de inclusión financiera más utilizados de modo que se puede utilizar de forma original o personalizarlo para apoyar directamente a las necesidades de sus clientes.

- **API**. La funcionalidad subyacente central necesaria por todas las instituciones de servicios financieros está totalmente expuesto y documentado como API REST desde la plataforma para que pueda crear rápidamente sus propios productos de calidad empresarial de inclusión financiera, aplicaciones móviles o nuevas soluciones en cualquier pila de tecnología.

Debido a que la mayoría de empresas dedicadas a las finanzas en especial las pequeñas compañías optan por hacer uso de este tipo de sistemas para llevar un control sobre sus servicios y clientes es totalmente viable usar una plataforma como lo es MIFOS, pero también se debe tener en cuenta que cada compañía tiene su propio sistema de servicios, es decir, aunque se debe cumplir ciertos parámetros ya establecidos generalmente es posible realizar ciertas modificaciones en estos planes, por lo cual dicha plataforma debe ser modificada a estos parámetros, un ejemplo de esto es el sistema de créditos que dependiendo la compañía los términos de *"crédito"* pueden variar, pero en general un crédito es una operación financiera donde una persona (acreedor) presta una cantidad determinada de dinero a otra persona (deudor), en la cual, este último se compromete a devolver la cantidad solicitada en el tiempo o plazo definido de acuerdo a las condiciones establecidas para dicho préstamo más los intereses devengados, seguros y costos asociados si los hubiera.

En resumen, se puede decir que por medio del sistema **MIFOS X** se brindara las herramientas necesarias y el conocimiento enfatizado en la herramienta a través de un proceso investigativo y haciendo uso de la ingeniería inversa para que a futuro esta herramienta pueda ser modificada según las necesidades que se presente sin inconveniente alguno, y sobre todo pueda brindar a sus clientes la satisfacción de un servicio eficiente y de buena calidad.[3]

2.2 REINGENIERÍA DE CÓDIGO

La reingeniería se comprende como el proceso mediante el cual se tiende a realizar mejoras y/o modificaciones de un software ya existente por lo que se hace uso de técnicas de ingeniería inversa y reestructuración de código. Para iniciar el proceso de reingeniería se debe utilizar un modelo que comúnmente es un modelo cíclico, aunque no se descarta modelos de tipo secuencial o lineal, es decir, puede ser que la ingeniería inversa (la comprensión del funcionamiento interno del software) tenga que producirse antes de que pueda comenzar la reestructuración del código.

La reingeniería de software también se puede definir como la modificación de un producto (software), o determinados componentes de este, usando técnicas de ingeniería inversa para la etapa de análisis, y para la etapa de reconstrucción herramientas específicas de tal manera que se oriente el cambio hacia mayores niveles de facilidad en cuanto a mantenimiento, reutilización, comprensión entre otras cualidades.

Es normal que se deseen realizar cambios en una aplicación que haya sido utilizada por un tiempo bastante considerable, esto ocurre debido a los

[3] COMUNIDAD MIFOS. Mifos Initiative. [agosto 2012]. United State. <http://mifos.org/> [Citado marzo 23 de 2017].

constantes cambios que se ven en los diferentes sectores lo que integra nuevas necesidades específicas de cada sector generando de esta forma cambios en la herramienta utilizada que logren cubrir estas nuevas necesidades. Lo que implica que cada vez que se genere un cambio se puedan llegar a producir efector no deseados, por lo que se hace necesario aplicar la reingeniería a la herramienta en cuestión para que de tal manera que tras los cambios realizados esta siga siendo de utilidad.

Los beneficios que se obtienen tras aplicar técnicas de reingeniería son los siguientes:

- Pueden reducir los riegos evolutivos de una organización.
- Puede ayudar a las organizaciones a recuperar sus inversiones en software.
- Puede hacer el software más fácilmente modificable
- Amplía las capacidades de las herramientas CASE
- Es un catalizador para la automatización del mantenimiento del software
- Puede actuar como catalizador para la aplicación de técnicas de inteligencia artificial para resolver problemas de reingeniería

2.2.1 Pasos de ingeniería inversa aplicadas a software. Los pasos se basan en tres métodos sencillos que son:[4]

- **Análisis de Inventario.** Es una buena técnica tener diversos inventarios al momento de desarrollar una herramienta ya que puede proporcionar información de vital importancia al momento de tomar decisiones. El inventario por simple que sea deberá contener información que proporcione una descripción detallada de la herramienta en cuestión. Es importante señalar que el inventario deberá actualizarse con regularidad, el estado de las aplicaciones puede cambiar en función del tiempo y, como resultado, cambiarán las prioridades para la reingeniería.
- **Restructuración de código.** Algunos sistemas heredados tienen una arquitectura de programa relativamente sólida, pero los módulos individuales han sido codificados de una forma que hace difícil comprenderlos, comprobarlos y mantenerlos. En estos casos, se puede reestructurar el código ubicado dentro de los módulos que no se lograron entender. Para llevar a cabo esta actividad, se analiza el código fuente mediante una herramienta de reestructuración, y entonces se reestructura el código. El código reestructurado resultante se revisa y se comprueba para asegurar que no se hayan incluido funciones indeseadas.
- **Restructuración de datos.** Un programa que posea una estructura de datos débil será difícil de adaptar y de mejorar. De hecho, para muchas

[4] RAMOS-ACOSTA. Óp. Cit,

aplicaciones, la arquitectura de datos tiene más que ver con la viabilidad a largo plazo del programa que el propio código fuente. A diferencia de la reestructuración de código, que se produce en un nivel relativamente bajo de abstracción, la estructuración de datos es una actividad de reingeniería a gran escala. En la mayoría de los casos, la reestructuración de datos comienza por una actividad de ingeniería inversa. La arquitectura de datos actual se analiza minuciosamente y se definen los modelos de datos necesarios. Se identifican los objetos de datos y atributos y, a continuación, se revisan las estructuras de datos a efectos de calidad.

Los pasos anteriores son fundamentales para la aplicación de técnicas de ingeniería inversa en un proyecto, y tras aplicarlos de manera adecuada el resultado tiene una alta probabilidad de que el porcentaje de satisfacción por parte del usuario sea alto generando así todos los beneficios vistos anteriormente.

2.2.2 Herramienta y análisis. La herramienta seleccionada para el presente proyecto es ObjectAid UML Explorer ya que es una herramienta cuyo resultado puede generar información sobre el código de vital importancia para obtener un mejor conocimiento de la herramienta en su parte técnica y al momento de realizar modificaciones sobre este pueda ser más óptimo.[5]

Por su parte ObjectAid UML Explorer es una herramienta ágil y ligera de visualización de código para el Eclipse IDE. Muestra el código fuente de Java y las bibliotecas en la clase UML en directo y los diagramas de secuencia que se actualizan automáticamente a medida que cambia el código. Es un proyecto que rige desde el año 2009 y cuenta con actualizaciones recientes.

La herramienta ha pasado por diversas versiones desde su salida las cuales son importante mencionarlas, cabe destacar que algunas versiones tienen mayor relevancia que otras:

Tabla 1. Descripción de cada versión de la herramienta ObjectAid UML (objectaid, 2017)

Versiones	
Fecha	**Cambio**
4 de febrero de 2012	La versión 1.0.7 tiene algunas correcciones de errores, los detalles son como de costumbre en la sección de descarga.

[5] HERRAMIENTA DE INGENIERÍA INVERSA. ObjectAid UML. [Enero de 2013]. Estados Unidos. < http://www.objectaid.com/>. [Citación febrero 20 de 2017],

26 de abril de 2012	La versión 1.0.8 hace que el menú contextual del Diagrama de Secuencias 'Add Called Operations' esté disponible en métodos estáticos. También es posible ocultar los estereotipos en los Diagramas de Clase. Los menús contextuales del Diagrama de Clases y Secuencias se han aplanado y, en algunos casos, han sido reordenados.
4 de junio de 2012	La versión 1.0.9 hace que ObjectAid sea compatible con la próxima versión de Eclipse 4.2 (Juno); Las pruebas se realizaron con 4.2-M6 y 4.2-M7. También 'Añadir operaciones llamadas' ahora utiliza objetos de diagrama existentes si su nombre corresponde a un campo o una variable en el código.
5 de agosto de 2012	La versión 1.0.10 mejora la detección de dependencias para los diagramas de clases y 'Añadir operaciones llamadas' para diagramas de secuencia. Este último usará ahora el método de prioridad más específico disponible en el contexto actual.
9 de noviembre de 2012	1.0.11 es una pequeña corrección de errores, principalmente para solucionar un problema con Eclipse 4.2.1. Además, ObjectAid se complace en anunciar el Add-On de Diagram, un plug-in comercial que trae características completamente nuevas que muchos de ustedes han estado pidiendo: notas de texto en diagramas, más formatos de exportación (SVG, PDF), la capacidad de ocultar asesores de propiedad, Colores diferentes para nodos de diagrama y mucho más.
19 de febrero de 2013	El nuevo complemento de diagrama ya está disponible en la versión 1.1.0.

	Trae características que muchos de ustedes han estado pidiendo: notas de texto, más formatos de exportación (SVG, PDF), la capacidad de ocultar accesores de propiedad, diferentes colores para nodos de diagrama y mucho más. Todas las licencias de Diagrama de Secuencia se han actualizado a la versión 1.1.x. Si la licencia adquirida se encuentra en modo en línea, su copia local de la licencia debe actualizarse automáticamente en Eclipse o cuando pulse el botón "Renovar" en las preferencias de ObjectAid. 28 de febrero de 2013 La versión 1.1.1 corrige varios problemas más pequeños, los detalles se encuentran en la sección de descarga.
23 de marzo de 2013	La versión 1.1.2 trae otra mejora para el Complemento de Diagrama: Una barra de herramientas que facilita el cambio de las opciones de visualización en las clases de los diagramas de clases. También corrige varios problemas más pequeños, los detalles están en la sección de descarga.
5 de junio de 2013	La versión 1.1.3 garantiza la compatibilidad con Eclipse 4.3.0 (Kepler) y corrige varios problemas (consulte la sección de descarga). También es posible ahora controlar la visualización de miembros estáticos en diagramas de clases, por ejemplo, desde la barra de herramientas del Diagrama Add-On.
1 de septiembre de 2013	Junto con muchas mejoras y correcciones de errores, la versión 1.1.4 trae una nueva barra de herramientas para el Add-On de Diagrama. Hace las acciones de

	alineación fácilmente disponibles sin tener que pasar por un menú contextual. También un nuevo menú contextual de Diagrama Add-On le permite agregar dependientes de un clasificador a un diagrama. Dependientes son clasificadores que son utilizados por el actual como variables, parámetros, clases internas anónimas, etc., pero no hay asociación, generalización, realización o relación de anidamiento. En el sitio web de ObjectAid, ahora es posible transferir licencias adquiridas a otra cuenta. Esta característica es útil para los revendedores. Les permite comprar licencias por cuenta propia y luego transferirlas a la cuenta ObjectAid de sus clientes. Otras entidades más grandes también deberían poder beneficiarse de una separación entre contabilidad y gestión de licencias.
1 de marzo de 2014	Después de un descanso ligeramente más largo para las mejoras internas necesarias, la versión 1.1.5 ya está disponible en la sección de descarga. Su principal novedad es la capacidad del Diagram Add-On de definir sus propias asociaciones a partir de parámetros de tipo genérico.
19 de mayo de 2014	La versión 1.1.6 permite la renovación de las licencias en línea en los casos en que un proxy local previamente lo previno. Esto sucede cuando el proxy utiliza un certificado autofirmado, lo que evita que el JDK realice una conexión SSL a Internet. El Complemento de Diagrama ahora le permite controlar qué nodo de diagrama de clases aparece en la parte superior, lo cual es útil cuando varios de ellos se superponen.
28 de octubre de 2014	La versión 1.1.7 trae más mejoras

	para el complemento de diagrama: Ahora puede ordenar las características de una clase alfabéticamente o dejarlas como están en el código. Las asociaciones se pueden mostrar como agregaciones o compuestos. Un ejemplo está aquí. También ofrecemos precios académicos ahora. Debido a un pequeño cambio en el embalaje, no es posible actualizar a la versión 1.1.7 del Diagrama de Clase de una versión anterior. Desinstale una versión anterior del diagrama de clases de ObjectAid siguiendo estos pasos: Vaya a Ayuda> Acerca de Eclipse> Detalles de la instalación. En la pestaña "Software instalado", seleccione "Diagrama de clase ObjectAid". Pulse 'Desinstalar ...' y luego 'Finalizar'. Pulse 'Reiniciar ahora' cuando se le solicite. Ahora está listo para instalar la versión 1.1.7.
2 de enero de 2015	Con la versión 1.1.8 ahora soportamos licencias con nodo bloqueado. Una licencia con nodo bloqueado puede ser utilizada por cualquier usuario en una computadora en particular. Cualquier licencia de usuario se puede convertir en una licencia con nodo bloqueado cuando inicie sesión en su cuenta de ObjectAid.
5 de noviembre de 2015	La versión 1.1.9 le brinda el enrutador Manhattan que mantiene todos los segmentos de relación tanto horizontales como verticales, independientemente de cómo mueva los puntos flexibles individuales. Además, el diagrama de secuencias ahora admite mensajes síncronos y

	asíncronos. Todos los mensajes existentes se sincronizarán y tendrán un triángulo completo en el extremo de destino. Puede hacerlos asíncronos con un menú contextual.
17 de junio de 2016	La versión 1.1.10 tiene algunas correcciones para la compatibilidad con Eclipse 4.6 (Neon). También trae la versión requerida de JDK a 8.0, que es igual que Eclipse 4.6; JDK 7 no ha recibido una actualización desde hace más de un año. Estamos felices de anunciar una actualización importante para el Diagrama de Secuencias en la versión 1.2.0: Apoyará fragmentos combinados que pueden ser ingeniería inversa de código fuente; Por supuesto también se pueden crear y mantener manualmente.
11 de septiembre de 2016	En la versión 1.1.11, el requisito de JDK 8.0 se ha eliminado de la versión 1.1.x. JDK 8.0 seguirá siendo necesario para la próxima versión 1.2 y más allá.
21 de noviembre de 2016	La versión 1.1.12 es una pequeña versión de corrección de errores para el enrutador de Manhattan.
30 de noviembre de 2016	La versión 1.1.13 es otra pequeña versión de corrección de errores para el enrutador de Manhattan.
20 de febrero de 2017	La versión 1.1.14 corrige algunos problemas relacionados con las imágenes automáticas.

Fuente: http://www.objectaid.com/

Las aplicaciones se reconstruyen utilizando un “motor de reingeniería” automatizado. En el motor se insertaría el programa antiguo, que lo analizaría, reestructuraría y después regeneraría la forma de exhibir los mejores aspectos de la calidad del software. Después de un espacio de tiempo corto, es probable que llegue a aparecer este “motor”, pero los fabricantes de CASE han presentado herramientas que proporcionan un subconjunto limitado de estas capacidades y que se enfrentan con dominios de aplicaciones específicas. Lo que es más importante, estas herramientas de reingeniería cada vez son más sofisticadas.

La Ingeniería directa, que se denomina también renovación o reclamación, no solamente recupera la información de diseño de un software ya existente, sino que, además, utiliza esta información en un esfuerzo por mejorar su calidad global. En la mayoría de los casos, el software procedente de una reingeniería vuelve a implementar la funcionalidad del sistema existente, y añade además nuevas funciones y/o mejora el rendimiento global.

2.3 ADMINISTRACIÓN DE CONOCIMIENTO

2.3.1 Ciclo de vida del conocimiento. Las compañías ingeniosas parten del conocimiento institucional para progresar con éxito en sus operaciones futuras. Documentan sus mejores prácticas, métodos y lecciones de la vida real eficaces para reutilizarlos en proyectos futuros. Esta es la piedra angular de la gestión del ciclo de vida del conocimiento (KLM): encontrar una forma para recopilar conocimiento y distribuirlo en organizaciones completas (tata technologies, pionera en gestión del conocimiento en el mundo).

El ciclo de vida del conocimiento parte de muchos conceptos que involucran los diferentes procesos de desarrollo tanto de producto o servicios, para cual la compañía está desarrollando su ciclo de vida del conocimiento, por ejemplo una empresa como COCACOLA, debe tener un sistema de gestión del conocimiento, para los múltiples líneas de productos que la compañía, presta, con el fin de que ese conocimiento ya aprendido “lecciones aprendidas”, se pueda gestionar de una manera más eficiente, para que las personas que necesiten tener acceso a esa información, puedan tener un pleno conocimiento de cómo se llevaron a cabo la elaboración de dichos productos.

MIFOS X es una plataforma financiera capaz de soportar múltiples servicios financieros, esta aplicación Fue diseñada por una Compañía desarrolladora de Software Libre llamada Fundación Grameen, inspiradas por una labor social y financiera, que es Banca rizar los países o regiones que no tienen acceso al sector financiero con el fin de darles la oportunidad de banca rizar estas comunidades y así puedan realizar sus proyectos y cubrir sus necesidades.

MIFOS por ser una plataforma libre, solo posee funciones básicas para soportar las operaciones más básicas del sector financiero, por ende dependiendo de la necesidad del negocio y las leyes gubernamentales que rigen en cada país del mundo, se debe adecuar la herramienta a las necesidades de la empresa a utilizarla, por ser una aplicación desarrollada bajo una licencia de código libre, su código fuente, alguna documentación, y una comunidad que apoya la iniciativa están disponibles para adecuar la herramienta dependiendo de la necesidad del cliente.

Lamentablemente la información documental sobre la herramienta, es muy escasa, lo cual dificulta múltiples necesidades de conocimiento y diseño de la misma. Por ende, vemos aplicar una metodología de gestión de conocimiento, para indagar más sobre cómo, a partir de la información disponible sobre la herramienta, un usuario deseoso de adaptar esta herramienta dentro de su organización, podría realizarlo, Por medio de los siguientes pasos a seguir:

- **Catálogo de información disponible.** Un catálogo comúnmente se conoce como una herramienta de recolección de información capas de clasificar y mostrar a mayor rango las características de productos y servicios ofrecidos por una compañía en general, en este caso en específico la utilizaremos para clasificar y recolectar en un solo punto la información disponible sobre MIFOS, entre ellas encontraremos:
- Información documental Disponible sobre MIFOS.
- Código fuente.
- Comunidades que la patrocinan.
- Bloc.
- Foros.
- Redes sociales.
- **Identificar las necesidades presentadas por los usuarios.** En esta fase lo que se pretende es identificar por medio de foros de comunidades oficiales de MIFOS, las necesidades que se presentaron a lo largo del desarrollo, implementación y adaptación de la herramienta a las necesidades de los usuarios, en ellas encontraremos información de vital importancia como lo son:
  Posibles fallos de ejecución.
  Complementos propuestos por la comunidad para mejorar el rendimiento.
  Creación de módulos adicionales para incrementar la eficiencia operacional de la plataforma.
  Dudas con respecto al funcionamiento.
- **Desarrollar modelos de anotaciones para cada necesidad.** Un modelo de anotaciones nos permite tener una visión más clara de cómo podríamos por medio de decisiones tomadas por parte del motivador o motivadores, darle solución a necesidades inmediatas o futuras.
- El modelo se compone de los siguientes componentes.
- Motivador o Motivadores.
- Soluciones alternativas.
- Evaluación de esas soluciones.
- Se toma una decisión como solución a la necesidad propuesta.

- **Diseño de un modelo de decisiones.** El diseño de un modelo de decisiones, implica tener identificadas, clasificadas y tomadas, las soluciones de la decisiones que se tomaron, en este modelo lo que se

pretende es relacionar las decisiones tomadas, con el fin de construir un arquitectura de decisiones, si en el futuro se requiere de estas decisiones que ya se habían modelado, a partir de ese conocimiento adquirido poder dar solución a futuras necesidades que se presente durante y después de terminado el ciclo de vida de desarrollo del software.

2.3.2 Decisiones de arquitectura de software como conocimiento. Las decisiones de software como conocimiento recolectado, son la materia prima en investigación en **Administración del conocimiento,** si se tiene una base de datos completa, con un buen modelo de administración de los datos y sobre todo una relación ligada al enfoque principal de un Modelo de Decisiones que es Diagramas de anotaciones.[6]

2.3.2.1 Modelo de anotaciones. Gran parte del éxito de recuperación de toma de decisiones, es un diseño apropiado de modelos de decisiones. El diseño es una decisión, en el que los diseñadores ponderan sus preocupaciones y varias soluciones para el final escoger la más adecuada para solucionar la necesidad que se presenta en este caso, donde los diseñadores de los modelos de decisión, plantean una gran cantidad soluciones posibles pero al final solo una va a prevalecer entre todas ellas, en un ejercicio juicioso y de análisis pertinente, esta parte del proceso es de vital importancia ya que a partir de las decisiones que tomemos, repercutirán en la implementación de la solución final.

Las decisiones y sus razones por lo general no están documentadas Porque la captura de decisiones de diseño se considera intrusiva, tedioso y costoso. Las decisiones que toma el diseñador en el momento quedan solamente en su cabeza y desaparece con el tiempo si no se documenta.[3]

2.3.3 Recuperación de conocimiento. A partir del desarrollo de los modelos de anotación, el paso a seguir es cómo podemos relacionar estas decisiones en un modelo de decisiones, para así tener una visión más clara de cuales fueron los requerimientos y sus respectivas soluciones.

Es muy importante que una vez se tome una decisión con respecto a una necesidad planteada esta se plasme en un formato con las siguientes características como mostraremos a continuación como ejemplo:

[6] PEDRAZA-GARCIA, Gilberto, Astudillo, Hernán, Correal Darío. DVIA: Understanding how software architects make decisions in design meetings. In Proceedings of the 2015 European Conference on Software Architecture Workshops (ECSAW '15). ACM, New York, NY, USA. Article 51, 7 pages. 2015. [Citado febrero 15 2017].

Tabla 2. Formato descripción de captura de a anotación, para llevar a cabo la construcción del diagrama de anotaciones.[7]

Anotación	**Descripción**
problema	Representa la motivación, necesidad, o requerimientos al cual se desea dar solución.
Solicitud de aclaración	Es la aclaración, observación o corrección acerca de los aspectos descritos en las anotaciones cuando se requieren una aclaración.
Explicación	Es la respuesta a las solicitudes de aclaración.
Orientación	Es la contribución a las acercamientos, indicaciones o sugerencias acerca de cómo se resolverán una preocupación o problema motivado en la discusión.
Orientación obligatoria	Es una pauta de opinión para resolver preocupaciones.
Evaluación	Es un juicio, acerca del impacto que tendrían las orientaciones expresadas.
Acuerdo	Representa el consenso de aceptar una orientación que satisfaga la preocupación generada a partir de la discusión.
desacuerdo	Expresa el rechazo que se generó, a partir de la orientación la cual pretendió satisfacer las preocupaciones.
Decisión	Representa un enfoque que ya ha sido aceptado

Fuente: PEDRAZA-GARCIA, Gilberto, Correal Darío, Oscar González Rojas, Guillermo Beltrán. MONO+KM: Knowledge Management in Collaborative Project Development: MONO+KM: Administración de Conocimiento en el Manejo de Proyectos Colaborativos, página 8-9. 2015.

Una vez se haya descrito las anotaciones con respecto a las necesidades planteadas, el paso a seguir comenzar la construcción del modelo de anotaciones, con sus respectivas entidades, esta parte del proceso se hace con el fin de relacionar las necesidades que se plantearon en los formatos de descripción, los motivadores de esta necesidad, las decisiones y evaluaciones

[7] PEDRAZA-GARCIA, Gilberto, Correal Darío, Oscar González Rojas, Guillermo Beltrán. MONO+KM: Knowledge Management in Collaborative Project Development: MONO+KM: Administración de Conocimiento en el Manejo de Proyectos Colaborativos, página 8-9. 2015. [Citado Febrero 16 de 2017].

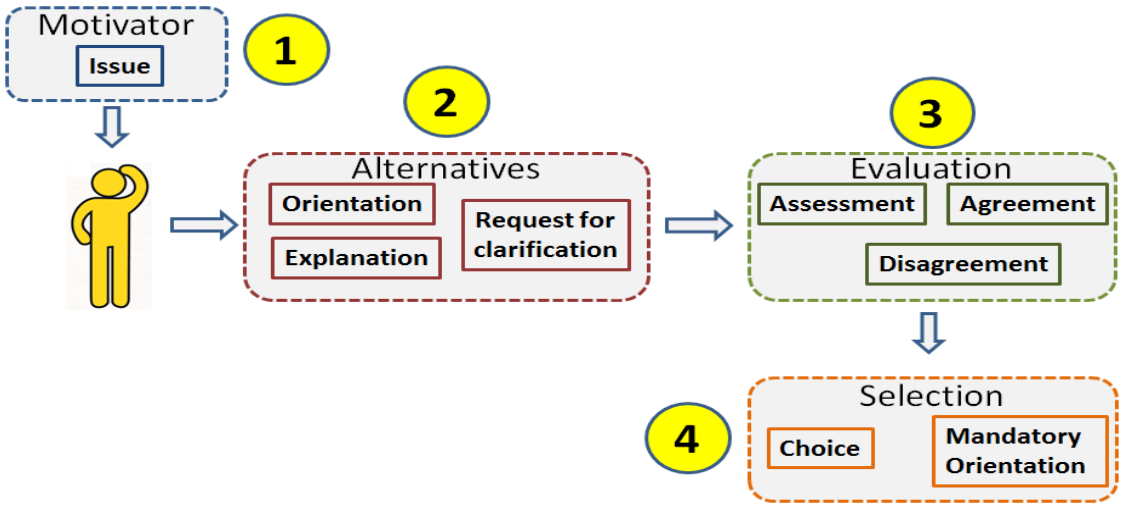
que se llevaron a cabo para darle una solución adecuada, y para esto se utilizara un algoritmo llamado "**Algoritmo de diferenciación de decisión**". [8]

2.3.3.1 Algoritmo de diferenciación de decisión. El objetivo del algoritmo es abstraer un conjunto de cuestiones de decisión de alto nivel y asignar las anotaciones recogidas durante la etapa anterior a cada elemento de decisión. El algoritmo descrito a continuación se basa en el GTM:

- Identificación de conceptos relevantes (codificación abierta).
  A través de un análisis de frecuencia de ocurrencia de términos individuales en los textos que comprenden anotaciones, se identifican términos candidatos a resaltar a conceptos.
  A través de un proceso de comparación constante los términos están asociados con contextos, y se establece el siguiente nivel de abstracción: Conceptos[8].
  Definición de asociaciones concepto-contexto. A continuación, se presentan conceptos interrelacionados para generar asociaciones concepto-contexto. Este proceso de comparación constante puede elevar los términos de prioridad a un nivel conceptual. Estas asociaciones se analizan semánticamente para dejar sólo aquellas que mejor explican un conjunto de anotaciones. Estas asociaciones concepto-contexto son categorías en un tercer nivel de abstracción.
- Definición de jerarquías de asociación (codificación axial). Las asociaciones concepto-contexto se clasifican por el tipo de proceso al que pertenecen, lo que permite establecer jerarquías entre asociaciones por nivel de abstracción.
- Refinamiento de la decisión sobre conceptos fundamentales (Codificación selectiva). Cada tema de decisión seleccionado como acto para hacer parte de la jerarquía se asigna a las anotaciones correspondientes. Para lograr esto, se realizan los siguientes pasos, como se muestra en la Figura abajo:
  Asigne cada anotación de Issue a un motivador de decisión.
  Asigne cada orientación, solicitud de aclaración y anotación de explicación a una solución alternativa de decisión.
  Asignar cada anotación de Evaluación, Acuerdo y Desacuerdo a una decisión de evaluación alternativa.
  Asigne cada anotación de Orientación y selecciona una Opción de decisión.

Figura 1 Asignar anotaciones a los componentes de decisión.

[8]PEDRAZA-GARCIA, Gilberto. MONO .Óp. pág. 9

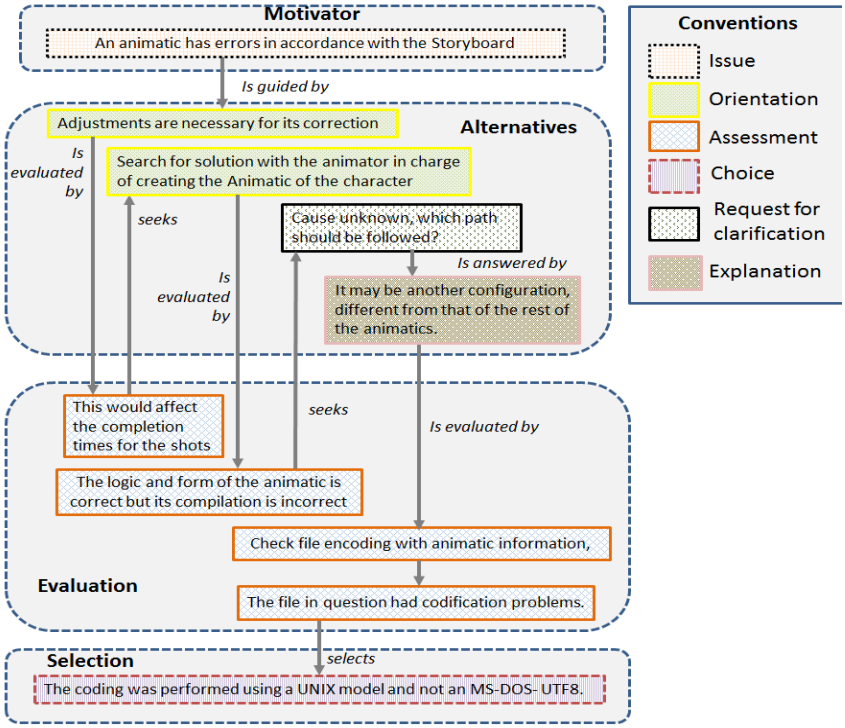


Fuente: PEDRAZA-GARCIA, Gilberto, Correal Darío, Oscar González Rojas, Guillermo Beltrán. MONO+KM: Knowledge Management in Collaborative Project Development: MONO+KM: Administración de Conocimiento en el Manejo de Proyectos Colaborativos, página 10. 2015

Después de aplicar el algoritmo de diferenciación de decisión, pasamos al siguiente paso que es el **Diseño del diagrama de decisiones**[9]**,** que compone la tipificación de todas las entidades involucradas dentro del modelo de decisión como lo mostraremos a continuación:

Figura 2. Ejemplo del diseño de un diagrama de Decisiones, con sus respectivas conversiones e incluido su diagrama de Anotaciones.

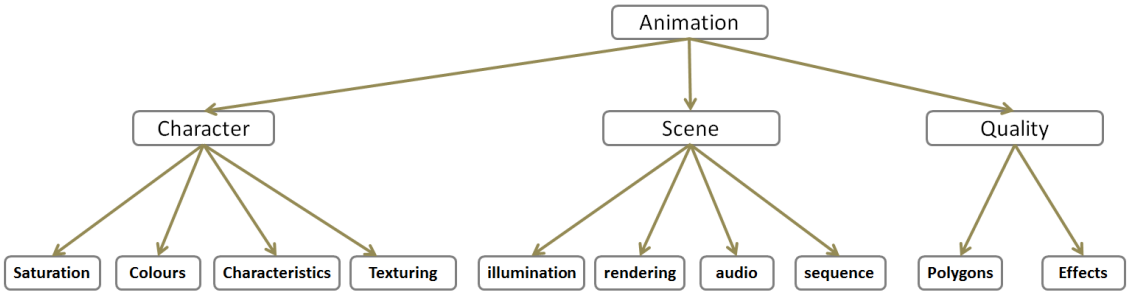
[9]PEDRAZA-GARCIA, Gilberto. MONO .Óp. cit. pág. 18



Fuente: PEDRAZA-GARCIA, Gilberto, Correal Darío, Oscar González Rojas, Guillermo Beltrán. MONO+KM: Knowledge Management in Collaborative Project Development: MONO+KM: Administración de Conocimiento en el Manejo de Proyectos Colaborativos, página 18. 2015.

Una vez se haya implementado todos los diagramas de decisión, se deben jerarquizar para conocer la importancia de cada una de las decisiones tomadas a partir de realizar la recuperación de conocimiento de toma de decisiones como se presentará a continuación:

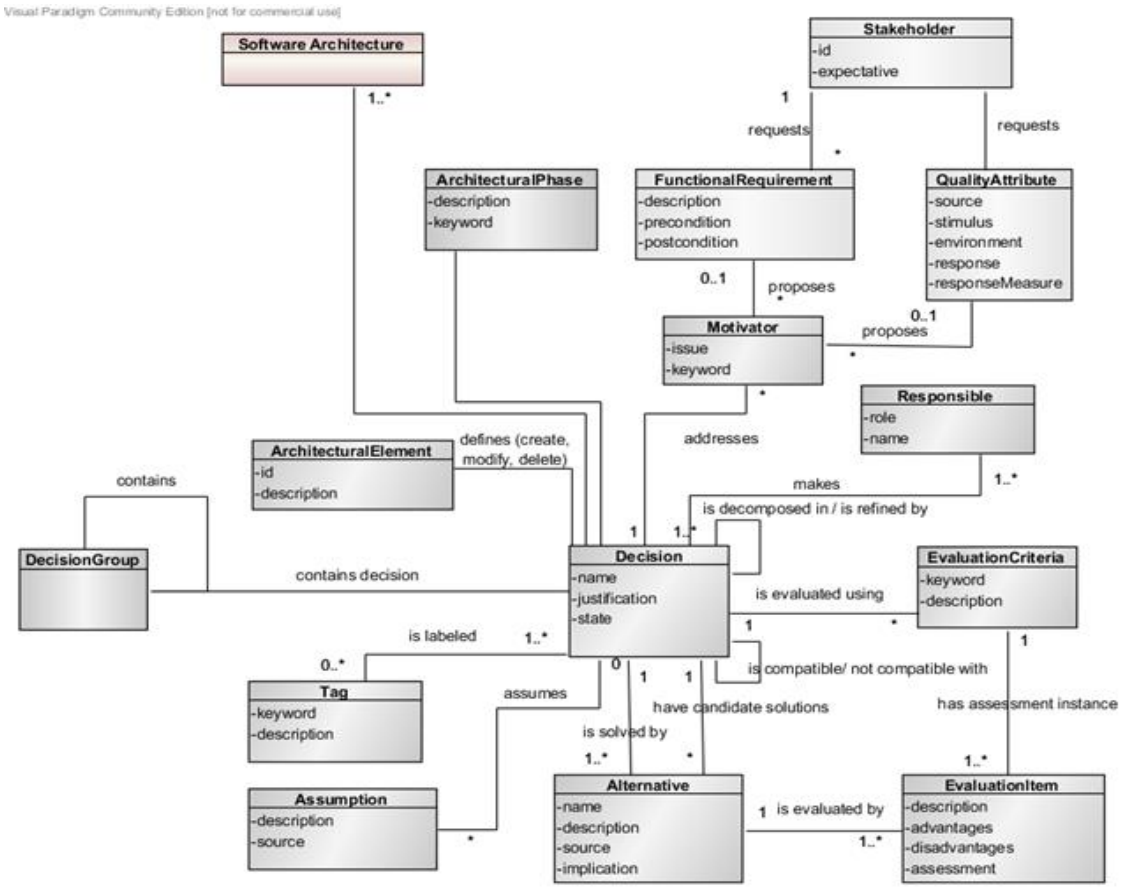
Figura 3. Ejemplo del diseño de un diagrama de Decisiones, con sus respectivas conversiones e incluido su diagrama de Anotaciones.



Fuente: PEDRAZA-GARCIA, Gilberto, Correal Darío, Oscar González Rojas, Guillermo Beltrán. MONO+KM: Knowledge Management in Collaborative Project Development: MONO+KM: Administración de Conocimiento en el Manejo de Proyectos Colaborativos, página 18. 2015.

2.3.4 Representación de decisiones de diseño

Figura 4. Ejemplo del diseño modelo de toma de decisiones.



Fuente: PEDRAZA-GARCIA, Gilberto, Correal Darío, Oscar González Rojas, Guillermo Beltrán. MONO+KM: Knowledge Management in Collaborative Project Development: MONO+KM: Administración de Conocimiento en el Manejo de Proyectos Colaborativos, página 8. 2015.

2.4 MARCO TECNOLÓGICO

2.4.1 Wiki. Es el nombre que recibe un sitio web, cuyas páginas pueden ser editadas directamente desde el navegador, donde los usuarios crean, modifican o eliminan contenidos que, generalmente, comparten. No tiene por qué ser necesariamente un sitio en la web, puesto que hay wikis instalables para uso en el escritorio de un computador personal, o portables en un llavero usb que llevan un entorno LAMP, como por ejemplo XAMPP.

Los textos o «páginas wiki» tienen títulos únicos. Si se escribe el título de una página wiki en algún sitio del wiki entre dobles corchetes ([[Título de la página]]) esta palabra se convierte en un «enlace web» a la página correspondiente. De este modo, en una página sobre «alpinismo» puede haber una palabra como «piolet» o «brújula» que esté marcada como palabra perteneciente a un título de página wiki. La mayor parte de las implementaciones de wikis indican en el localizador de recursos uniforme (URL) de la página el propio título de la página wiki (en Wikipedia ocurre así: <https://es.wikipedia.org/wiki/Alpinismo> es el URL de la página wiki Alpinismo), facilitando el uso y comprensibilidad del enlace fuera del propio sitio web. Además, esto permite formar en muchas ocasiones una coherencia terminológica, generando una ordenación natural del contenido.

Las aplicaciones de mayor peso y a la que le debe su mayor fama hasta el momento ha sido la creación de enciclopedias colectivas, género al que pertenece Wikipedia. Existen muchas otras aplicaciones más cercanas a la coordinación de informaciones y acciones, o la puesta en común de conocimientos o textos dentro de grupos.

La mayor parte de los wikis actuales conservan un historial de cambios que permite recuperar fácilmente cualquier estado anterior y ver qué usuario hizo cada cambio, lo cual facilita el mantenimiento conjunto y el control de usuarios nocivos. Habitualmente, sin necesidad de una revisión previa, se actualiza el contenido que muestra la página wiki editada.

2.4.1.1 MediaWiki. Es un software para wikis libre programado en el lenguaje PHP. Es el software usado por Wikipedia y otros proyectos de la Fundación Wikimedia (Wikcionario, Wikilibros, etc). Ha tenido una gran expansión desde 2005, existiendo un gran número de wikis basados en este software que no mantienen relación con dicha fundación, aunque sí comparten la idea de la generación de contenidos de manera colaborativa. Se encuentra bajo la licencia de software GNU General Public License.

MediaWiki puede ser instalado en los servidores web Apache, Internet Information Services, Cherokee (servidor web), Hiawatha, LiteSpeed Standard (necesita la extensión Math), nginx, y lighttpd y puede usar como motor de base de datos MySQL/MariaDB, PostgreSQL y SQLite.

También se llama MediaWiki al espacio de nombres (ver más abajo) donde se hallan entre otras cosas los mensajes de su interfaz listos para su traducción a la lengua local de cada wiki, en caso de no estar todavía traducidos.

MediaWiki fue desarrollado originalmente para Wikipedia por Magnus Manske, con el fin de sustituir a UseModWiki como motor del wiki (al que los

colaboradores de Wikipedia llamaron "Fase I"). A la primera versión se la llamaba, simplemente "software de Wikipedia fase II".

A mediados del 2002 el programa fue reescrito y mejorado, dando lugar a la llamada "fase III", y ha seguido desarrollándose desde entonces a partir de ese código. El 29 de agosto de 2003 se bautizó al programa, hasta ese momento sin un nombre oficial, como "MediaWiki", un juego de palabras con el nombre de la Fundación Wikimedia, que patrocina su desarrollo. La primera versión con este nombre se llamó, entonces, "MediaWiki-stable 20030829". Se empezó entonces a pensar las nuevas versiones pensando en posibles usuarios ajenos a Wikipedia, mejorando especialmente en aspectos como la instalación del software.

El nombre "MediaWiki" es criticado en ocasiones por ser fácil de confundir con el de la fundación por parte de gente ajena a Wikipedia.

Las características son:

- A diferencia de los wikis clásicos, los nombres de las páginas no tienen por qué estar en CamelCase, lo que permite tener nombres más naturales.
- Espacios de nombres: permiten separar páginas de distintos tipos. Así, se puede tener un espacio de nombres para artículos, otro para plantillas, otro para debates, etc. que el software trata de distinta forma.
- Páginas de discusión: cada página del wiki tiene una página de discusión propia, dedicada a hablar de su mejora u otros fines.
- Soporte de TeX, para visualizar fórmulas matemáticas. Las fórmulas pueden mostrarse de varias formas, según las capacidades del navegador.
- Listas de seguimiento, de tal forma que cada usuario pueda seguir los cambios en los artículos de su interés.
- Sistema de *plugins* que permite extender fácilmente el software. Los plugins instalados se listan automáticamente en "Páginas especiales".
- Capacidad de bloquear temporalmente usuarios o páginas.
- Soporte de plantillas personalizadas con parámetros.
- Creación de líneas de tiempos a través de código wiki.
- Sistema de categorías jerárquico, que permite crear listados de artículos o de *thumbnails* de imágenes.
- Admite varios niveles de usuario, así como la posibilidad de que sólo los usuarios registrados puedan editar, o de impedir el registro de más usuarios. Así, puede utilizarse como sistema de gestión de contenidos o como groupware.
- Soporte para almacenamiento de memoria virtual o caché, también conocidos como memcached y el sistema de caché Squid.
- Pieles o máscaras ("skins") personalizables por cada usuario.

2.4.1.2 Semantic MediaWiki. Es una extensión para MediaWiki, que permite anotar datos semánticos dentro de páginas wiki, convirtiendo así a una wiki que incorpore la extensión en una wiki semántica. Los datos que han sido codificados pueden ser utilizados en búsquedas semánticas, usados para agregación de páginas, mostrados en formatos como mapas, calendarios y gráficos, y exportados al mundo exterior a través de formatos como RDF y CSV.

Cada anotación semántica dentro de SMW es una "propiedad" que conecta a la página en la que reside con alguna otra pieza de datos, ya sea otra página o un valor de algún tipo, usando triplas de la forma "sujeto, predicado, objeto".

A modo de ejemplo, una página sobre Alemania podría tener, codificado dentro de ella, el hecho de que su capital es Berlín. En la página "Germany" (Alemania en inglés), la sintaxis sería la siguiente:

Figura 5.Ejemplo de cómo crear una consulta en una wiki semántica.

```
... la capital es [[Has capital::Berlin]] ...
```

Fuente*:* Wikipedia enciclopedia libre.

Que es semánticamente equivalente a la declaración de "Germany" "Has capital" "Berlin" ("Alemania", "Tiene capital", "Berlín"). En este ejemplo, la página "Germany" es el *sujeto*, "Has capital" es el *predicado* y "Berlin" es el *objeto* al cual el enlace semántico está apuntando.

Sin embargo, la forma más común de almacenar datos dentro de Semantic MediaWiki es a través de plantillas de MediaWiki que en sí mismas contienen el marcado SMW necesario. Para este ejemplo, la página "Germany" podría contener una llamada a una plantilla llamada "Country" (país en inglés), que se parecía a esto:

Figura 6. Ejemplo de cómo almacenar datos dentro de una wiki semántica a través de una plantilla.

```
{{Country
...
|Capital=Berlin
...
}}
```

Fuente: Wikipedia enciclopedia libre.

La plantilla "Country" se encargaría de almacenar el valor del parámetro "Capital" utilizando la propiedad "Has capital". La plantilla también manejaría la presentación de los datos. Los desarrolladores de Semantic MediaWiki han estimado que el 99% los datos de SMW se almacenan de esta manera.

Semantic MediaWiki también tiene sus propias herramientas de consulta en línea. Por ejemplo, si las páginas sobre los países almacenasen información adicional como datos de población, una consulta podría ser añadida a una página que muestra una lista de todos los países con una población de más de 50 millones de habitantes, junto con su capital; y Alemania, aparecería en esa lista, con Berlín a su lado.

2.4.2 SparQL. SPARQL Protocol And RDF Query Language o también conocido como solo SPARQL es un lenguaje utilizado para la consulta de grafos RDF es de vital importancia en el uso de web semántica ya que las consultas realizadas dentro de esta se hacen por medio de dicho lenguaje que tiene un comportamiento similar a otros gestores de bases de datos.
Al igual que en otros sistemas de gestión de bases de datos como por ejemplo SQL, es importante saber la diferencia entre lenguaje de consultas, el almacenamiento de datos y la recuperación de datos, debido a esta diferencia se desarrollan múltiples implementaciones del lenguaje SPARQL que generalmente se encuentra relacionados con entornos de desarrollo y plataformas tecnológicas.

Inicialmente el lenguaje SPARQL solo integra funciones para la recuperación de sentencias RDF, sin no obstante algunas sentencias tienen como función operaciones para el mantenimiento, dichas operaciones comúnmente son crear, modificar y borrar datos e información de escala mayor.

2.4.2.1 RDF. También conocido como Resource Description Framework (Marco de descripción de recursos) es una serie de especificaciones que se desarrolló con el objetivo de usarse como modelo de datos para metadatos.
Actualmente es usado como un método general para la descripción conceptual o modelado de datos y/o información que se implementa en recursos web, por lo cual dispone de una variedad significativa de notaciones de sintaxis y formatos de serialización de datos.

2.4.2.2 SPARQL en Semantic MediaWiki. Dado que las bases de datos Semantic MediaWiki 1.6.0 RDF pueden usarse para almacenar datos que luego ofrecen servicios web SPARQL para consultar el wiki. Este soporte se mejoró sustancialmente con Semantic MediaWiki 2.0 y comenzando con Semantic MediaWiki 2.3.0. Las bases de datos RDF se consideran igualmente utilizables como bases de datos SQL.

3. DISEÑO METODOLÓGICO

3.1 SISTEMA DE HIPÓTESIS

3.1.1 Hipótesis investigativa. El desarrollo de una investigación basada en diferentes procesos de reingeniería a una herramienta desarrollada para el sector de microfinanzas llamada MIFOS X, la cual cuenta con su respectivo código fuente basado en una versión básica, así como diversos tutoriales, repositorios, wikis y una comunidad que ayudará a guiar el proceso y de esta manera completar los resultados deseados en la investigación.

3.1.2 Hipótesis específicas. La aplicación de métodos investigativos con el fin de realizar procesos de reingeniería en la herramienta facilita el entendimiento de esta misma, lo cual conlleva a resultados con un mayor porcentaje de satisfacción a la hora de realizar cambios en la herramienta según la necesidad, dicho de otra manera, se facilita el desarrollo de la herramienta y se proporciona el conocimiento teórico necesario.

Tras los resultados obtenidos en esta investigación se afirma que en varios factores importantes como lo son: tiempo, costos, conocimiento, entre otros se reducirán notablemente generando de esta manera una ayuda significativa en proyectos que se basen en la herramienta de microfinanzas MIFOS X.

3.1.3 Hipótesis nulas. La aplicación de métodos investigativos con el fin de realizar procesos de reingeniería en la herramienta no facilita el entendimiento de esta misma, lo cual conlleva a resultados con un menor porcentaje de satisfacción a la hora de realizar cambios en la herramienta según la necesidad, dicho de otra manera, no se facilita el desarrollo de la herramienta y no se proporciona el conocimiento teórico necesario.
Tras los resultados obtenidos en esta investigación se afirma que en varios factores importantes como lo son: tiempo, costos, conocimiento, entre otros no se reducirán notablemente generando de esta manera una ayuda poco significativa en proyectos que se basen en la herramienta de microfinanzas MIFOS X.

3.2 SISTEMA DE VARIABLES

3.2.1 Variables independientes. Desarrollo investigativo que genera procesos de ingeniería inversa y recuperación de información de la herramienta de microfinanzas MIFOS X.

En la presente investigación se considera como variable independiente el proceso de reingeniería o ingeniería inversa al cual es sometido una herramienta desarrollada para el sector de microfinanzas, debido a que se

pretende; brindar toda la información recopilada de los diferentes medios en base a la herramienta con el fin de que tanto el uso como el desarrollo de esta misma sea entendible a un nivel de facilidad mayor, y de esta misma manera dando a conocer los beneficios que puede brindar esta herramienta.

Se tomó como base investigaciones realizadas con un fin similar para ser más exacto la investigación MONO+KM.

3.2.1.1 MONO+KM. En la dinámica de la gestión de proyectos colaborativos las organizaciones participantes hacen grandes esfuerzos y aportan recursos técnicos, tecnológicos y humanos para conseguir un producto que difícilmente podrán desarrollar de forma individual. Aunque existen herramientas para integrar, controlar y gestionar procesos en este tipo de proyectos, no es común encontrar soporte tecnológico para gestionar el conocimiento generado durante su ejecución. Por lo general este conocimiento hace parte de la experiencia de los participantes, pero no se recupera, no se documenta, ni se aprovecha a nivel organizacional perdiendo un importante activo. En este estudio proponemos una técnica en la que se aplica un enfoque de administración de conocimiento a la gestión de proyectos colaborativos y donde el conocimiento es expresado en términos de decisiones. Esto se logra a partir del análisis de las interacciones verbales que se dan entre los participantes en este tipo de proyectos, la identificación y recuperación de decisiones mediante técnicas de Grounded Theory Method (GTM) y la especificación de un conjunto de escenarios concretos de uso. La técnica fue aplicada en MONO, un framework para integración, control y optimización de procesos de producción de contenidos digitales en la que trabajan colaborativamente empresas de la industria creativa. El estudio provee un modelo de anotaciones que sin ser intrusivo permite la recuperación y estructuración de conocimiento expresado como decisiones, haciendo viable su replicación en otros dominios[10].

Así mismo se puede basar la investigación realizada en el modelo DVIA.

3.2.1.2 DVIA. Gran parte del éxito de un proyecto de sistema de software es el diseño apropiado de la arquitectura del software. El diseño es una decisión, en el que los arquitectos ponderan sus preocupaciones Sobre las necesidades de funcionalidad y calidad expresadas por los Interesados. Así, las reuniones de diseño de arquitectura de software Son casos en los que los arquitectos discuten problemas de diseño y Decisiones sobre los elementos significativos de un sistema de software.

Tales decisiones pueden estar motivadas por preocupaciones, problemas y necesidades Que el equipo arquitectónico resuelve analizando y evaluar un conjunto de posibles soluciones, priorizar los criterios y elige una alternativa.

[10]PEDRAZA-GARCIA, Gilberto. MONO .Óp. pág. 1

Aunque los diseñadores conocen la importancia y las ventajas de las decisiones de diseño, las razones por las cuales estas no se documentan es que este tipo de técnicas se consideran intrusivas, aburridas y toman demasiado tiempo.

Perdiéndose así un valioso activo de información, el cual podría ser reciclado y reutilizado para futuras necesidades que se presenten dentro de una organización[11].

3.2.2 Variables dependientes. Las variables independientes constan de tres factores principales:

- **VD1**. Conocimiento sobre procesos de ingeniería inversa.
- **VD2**. Herramientas que ayuden en el proceso de ingeniería inversa de la herramienta.
- **VD3**. Decisiones de diseño que soporta la plataforma MIFOS X.

3.2.2.1 Mecanismo de Control*.* Aplicación de conocimiento de ingeniería inversa durante la investigación.

Someter la plataforma MIFOS X por diversas herramientas que contribuyan a la proporción de datos y/o información que ayuden a tener un mayor conocimiento de esta misma.

Recuperar decisiones que hayan sido tomadas al momento del desarrollo de la herramienta con el fin de plasmarlas en diagramas que sean entendibles para el usuario.

3.2.3 Variables intervinientes

3.2.3.1 Cantidad de información disponible de MIFOS X. La información actual que se tiene sobre la herramienta es confusa y en algunos casos nula, por lo cual se controla este ámbito uniendo toda la información encontrada de la herramienta en diferentes medios y presentando los datos más relevantes de la herramienta, generando de esta manera información de vital importancia al alcance del usuario.

3.2.3.2 Herramientas de reingeniería. El uso de herramientas de reingeniería para la herramienta **MIFOS X** es limitado ya que esta se encuentra desarrollada en modelo vista controlador (MVC) lo cual supone el uso de un framework en este caso AngularJS, conteniendo así en su parte FRONT diferentes lenguajes como lo son JavaScript, CSS, HTML, JSON, y en parte BACK se utiliza lenguaje JAVA, por lo cual es limitado el número de herramienta que puede

[11] PEDRAZA-GARCIA, Gilberto. Dvia. Óp. pág. 1.

soportar esta cantidad de lenguajes y la magnitud de código utilizado para el desarrollo de la herramienta.

Se controló este ámbito haciendo uso de herramientas para reingeniería de la parte BACK dividendo las clases que se usan para su composición de manera individual mostrando así exactamente la lógica de cada clase por medio de diagramas UML.

3.2.4 Operacionalización de variables

Tabla 3. Definición conceptual de variables independientes.

Variable independiente: desarrollo investigativo que genera procesos de ingeniería inversa y recuperación de información de la herramienta de microfinanzas **MIFOS X**.	
Definición conceptual. Contenido investigativo aplicando técnicas de ingeniería inversa para la recopilación de información sobre la herramienta **MIFOS X**.	Definición operacional: conjunto de actividades que incluyen el uso de herramientas externas para realizar el proceso de reingeniería sobre la herramienta, investigación de información recopilada para el conocimiento de la herramienta y las decisiones que se llevaron a cabo para este proyecto.

Fuente: Autores

Tabla 4. Definición conceptual de variables dependientes.

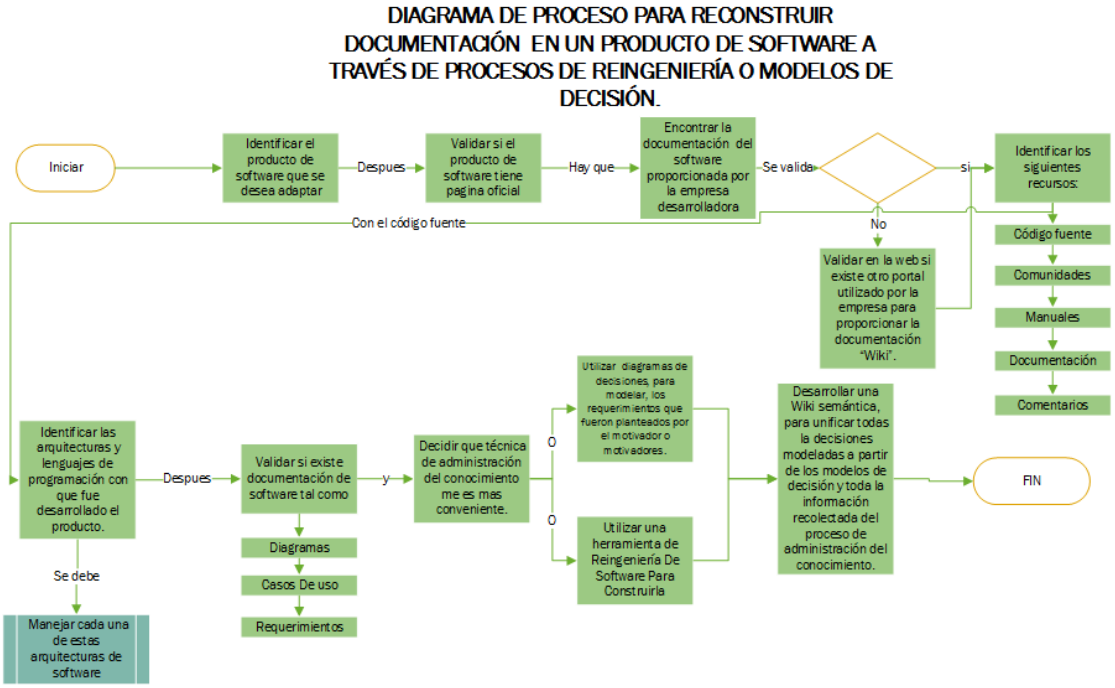
Variables dependientes	**Indicadores**
VD_1 Conocimiento sobre procesos de ingeniería inversa. **Definición Conceptual**. Hace referencia a los conocimientos en cuanto a procesos de ingeniería inversa se refiere que serán aplacados al presente proyecto.	**Conocimientos**. Aplicar los diferentes conocimientos adquiridos en cuando a ingeniería inversa se refiere, así como las diferentes metodologías que pueden hacer aportes importantes en el proceso investigativo.
Definición Operacional. Hace referencia a la información recopilada tras realizar diversos procesos de ingeniería inversa sobre la herramienta y que aporta dicha información para el conocimiento de esta misma.	**Ingeniaría Inversa**. Proceso el cual se usará en la finalidad del presente proyecto que consiste en descubrir cómo funciona la herramienta MIFOS X de cuyo código fuente se desea hacer modificaciones cubriendo nuevas necesidades o mejorando las ya existentes.
VD_2 Herramientas que ayuden en el proceso de ingeniería inversa de la	**Entorno**.

herramienta. **Definición Conceptual**. Hace referencia a las herramientas en concreto que se usarán para este proceso y los diferentes resultados que cada una proporcionará. **Definición Operacional**. Interpretación de los resultados que arrojaron las diferentes herramientas usadas para el proceso de ingeniería inversa y verificar que información puede ser de utilidad y descartar la que no.	Identificar el entorno de desarrollo de la herramienta, así como las posibilidades que este soporte para realizar el proceso de ingeniería inversa de la herramienta.
	Código. Hacer uso del código que se encuentra en los repositorios oficiales de la iniciativa **MIFOS** para someter este a diferentes pruebas de ingeniería inversa.
	Herramientas. Verificar que herramientas muestran resultados y de los resultados obtenido realizar un análisis detallo que seleccione los resultados que muestre información relevante.
VD_3 Decisiones de diseño que soporta la plataforma MIFOS X. **Definición Conceptual**. Hace referencia a la parte de la información recopilada donde se almacenan los tipos de decisiones que se tomaron a la hora del desarrollo y/o actualización de la herramienta en cuestión. **Definición Operacional**. Verificar la información referente a las decisiones de diseño implementadas en la herramienta, haciendo uso de estas como ejemplo para futuras tomas de decisión y proporcionando una ayuda informativa.	**Metodología**. Se indica lo diferentes métodos que se llevarán a cabo para la recopilación de decisiones de importancia significativa para el proyecto.
	Veracidad de Información. Realizar una meticulosa investigación con el fin de descartar información redundante o de poca importancia significativa que lo contrario de proporcionar conocimiento solo generaría más dudas o simplemente no responda a alguna inquietud.

Fuente: Autores

3.3 PROPUESTA DE SOLUCIÓN

Figura 7. Diagrama de proceso para reconstrucción Documental en un producto de software a través de procesos de reingeniería o modelos de decisión.



Fuente: Autores

3.3.1 Diagrama de proceso para Construir el proceso de diseño e implementación de un producto de software. Los productos son el resultado de múltiples esfuerzos de los distintos, ciclos de vida que se llevaron a cabo para su diseño, construcción, pruebas y finalmente su distribución al cliente final. La industria del Software no es la excepción, muchas de las compañías tienen procesos, alternos, que definen la calidad con la que fueron construidos sus productos, al final el resultado es siempre el mismo entregar un producto funcional, versátil, fácil de usar, y compatible para casi todos los sistemas operativos que existen en el planeta, sin embargo también existen organizaciones que se dedican a desarrollar herramientas que también llevan un ciclo de vida de desarrollo e implementación, pero a diferencia de las demás licencias esta tiene una particularidad y es que la conocida "propiedad intelectual" es de todos y para todos, por ende podríamos decir que es de todos pero también de nadie, a este tipo de productos los catalogamos como "Open Source" o licencia libre.

Básicamente este tipo de productos son dedicados para aquellas empresas que desean tener una independencia tecnológica plena, y no tener que estar sujetas a leyes de protección de derechos de Autores, es por eso que este tipo de productos deben tener a su disposición un completo set de herramientas con la capacidad de modificar, adecuar, innovar y compartir mejoras futuras sobre la Aplicación base.

Entre ellas encontramos las siguientes que son:

- Página oficial de la organización creadora del producto
- Encontrar la documentación proporcionado por el fabricante
- Validar si existe recursos de consulta tales como código fuente, comunidades, manuales, documentación y comentarios.
- En caso de que esta información anteriormente nombrada no se encuentre disponible en la página oficial, validar si se encuentra almacenada en otra página web, tales como wikis semánticas, foros o comunidades que apoyan la iniciativa.
- Identificar las arquitecturas con la que fue diseñada la aplicación “dominarlas o al menos entenderlas”.
- Validar si existe documentación de software disponible Diagramas, Casos De Uso, Requerimientos entre otros.
- Decidir que metodología de Administración del Conocimiento debo tomar para reconstruir decisiones de diseño.
- Si utilizamos diagramas de decisión, debemos identificar las anotaciones que se presentaron durante el ciclo de vida del software, con modelos de decisión.
- O utilizar Procesos de reingeniería de software, con alguna de las herramientas disponibles en el mercado que realizan este proceso.
- Desarrollar una wiki Semántica, para documentar toda la información recolectada mediante el proceso de recuperación de las decisiones, para que futuras generaciones puedan tener acceso a ellas, y puedan continuar contribuyendo y fomentando esta labor que muy pocos reconocen, que es librar al mundo del licenciamiento y ayudar a fomentar la creatividad e innovación en las tecnologías de la información.

3.3.2 Conjunto de información sin clasificar

3.3.2.1 Catalogación de toda la información disponible. La información es el activo más valioso que pueda poseer una organización, por ende, se debe tener una gestión y administración de toda esa información que se recolecta a través de los años, existen varias metodologías que nos permite clasificar, documentar y aprovechar todo ese conocimiento que se recolecto y aprovecho para nuevamente reutilizarlo a nuestro favor.

La clasificación de esta información es parte vital en un proceso de administración de conocimiento, ya que de que sirve tener disponible toda la información si no se aprovecha o reutiliza para beneficio propio:

¿Cómo Clasificar de manera eficiente y ordenada la información que poseemos?

- **Catálogo de información.** Un catálogo de información clasificada de toda la información que se tiene disponible para así logra un mejor entendimiento de la misma, comprender mejor la naturaleza de las

decisiones que se podrían a llegar a tomar si se entiende mejor el origen de donde vino.

Un catálogo de información está conformado por los siguientes tópicos.

- Versiones disponibles del producto.
- Características de cada versión.
- Documentación técnica disponible del producto.
- Código fuente “si está disponible”
- Foros.
- Comunidades.
- Redes sociales.

- **Versiones disponibles del producto.** Un producto de software debe tener disponibles múltiples versiones, las versiones son parte vital en el desarrollo y evolución en un productor de software ya que en este se evidencia los múltiples cambios y mejores que se han ido llevando a cabo a través del ciclo de vida del desarrollo del software como tal, a continuación, se mostrará un ejemplo de cómo debemos llevar a cabo una versión amiento en un catálogo de Información.

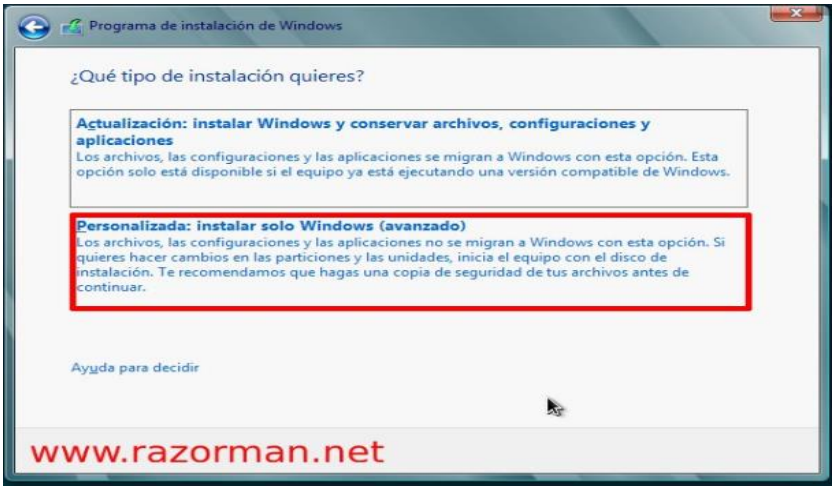
- **Características de cada versión.** Las características de cada versión es una fuente de información muy valiosa e importante, en el encontraremos las características, diferencias y descripción de cada versión de un producto de software y él porque se implementaron esos cambios.

3.3.2.2 Documentación técnica disponible.

- **Manuales de instalación de la aplicación base.** Un manual de instalación es un recurso, que nos permite conocer cómo se instalan y configuran cada dependencia o software necesario para que la herramienta base funcione adecuadamente.

Como se mostrará a continuación:

Figura 8 Ejemplo de un manual de instalación de un producto de software “Windows 8”.



Fuente: razorman.com instalación personalizada Windows 8.1. www.razorman.net

- **Manuales de instalación para modo desarrollador.** Un manual de instalación en modo desarrollador es muy distinto a un manual de instalación de la aplicación ejecutable, en este manual se describen de forma más técnica la instalación y configuración de las diferentes dependencias de una aplicación en uno o varios entornos de desarrollo.

Figura 9. Instalación de las diferentes herramientas de desarrollo de software disponibles en el mercado.

Fuente: Autores

- **Wikis.** Las wikis son una forma de documentación de información, en ellas encontramos información de vital importancia sobre funcionamiento e implementación de un producto, en caso de que la empresa desarrolladora lo requiera, ya que este este tipo de portales web son para compartir información y si el producto que deseamos documentar está sujeto bajo las leyes de protección de derechos de auto y propiedad intelectual, esta no sería la manera más adecuada de documentar el desarrollo de un producto de software. [6]

Figura 10. Ejemplo de cómo es la estructura de una wiki.

Fuente: AULAWIKI, wiki educativa para principiantes. www.aulawiki212.com

- **Código fuente.** El código fuente es el activo más valioso en un producto de software, dependiendo de la naturaleza de la metodología en que haya sido desarrollada una aplicación, estará disponible o no el código fuente de un producto de software a usuarios interesados en aportar mejoras en el mismo:

  Si fuera el caso donde el fabricante dejara disponible el código fuente a disposición de cualquier interesado, este se almacena por lo general en plataformas de desarrollo colaborativo como GITHUD.

Figura 11.Github es una plataforma de software colaborativa, capas de alojar proyectos de software y controlar versiona miento del mismo.

Fuente: página oficial de GITHUB. www.github.com.

- **Comunidades.** Una comunidad es un grupo de personas, con un interés en común en este caso desarrollo y mejoramiento de un producto de software, en ellas encontramos distintas formas de aprovechar esta interesante fuente de información.

- **Foros.** Los foros son sitios donde un grupo de personas con intereses en común, comparten su conocimiento sobre un tema en específico.

Figura 12.JavaHispano es una comunidad online presta servicios de certificación y comunidades de que apoyan en el principio de colaboración.

Fuente: página oficial de javahispano. www.javaHispano.com.

- **Redes sociales.** Las redes sociales cada día se involucran en el día a día de las personas y las organizaciones, tanto privadas como gubernamentales, son un medio de comunicación masivo con grandes expectativas.

  En ellas también podremos encontrar información de interés, como comunidades, enlaces, noticias referentes a las últimas versiones lanzadas de productos tecnológicos y sobre todo siempre están a la vanguardia.

Figura 13. Las redes sociales también son una herramienta muy poderosa a la hora de apoyar una causa, en este caso JavaHispano también tiene su red social.

Fuente: Facebook de javahispano. www.facebook.com.

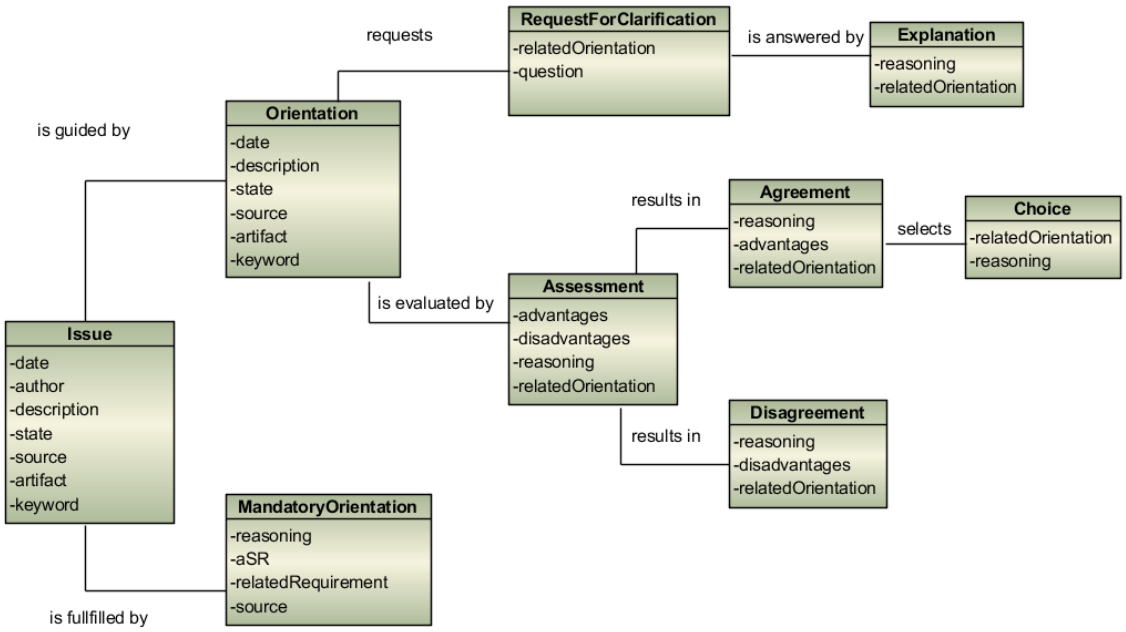
3.3.2 Instancia del modelo de anotaciones. Después de que se haya diseñado el catálogo de información lo siguiente, es localizar necesidades que se hayan presentado durante la ejecución del proyecto, esto por lo generar es manifestado por los usuarios por medio de los foros, comunidades, wikis entre otras.

Al localizar los requerimientos que se presentaron durante y después de la implementación del desarrollo del producto, pasamos a un ciclo llamado **diseño de modelos de anotación.**

3.3.2.1 Modelos de anotación. Un diagrama de anotación, es la interrelación que tiene los usuarios con sus necesidades, en el también encontraremos que a partir de esas necesidades que surgieron cuando se planteó, también surgieron ya sea por el mismo motivador u otras personas pertenecientes a la comunidad, las posibles soluciones y lo que se requiere para poder implementarlas, hay que tener en cuenta que al final se escoge una solución a necesidad plateada, a ella la llamaremos **decisión seleccionada**.

Si se desea saber cómo debemos diseñar un modelo de anotación por favor remitirse al capítulo 2 en el numeral de **Recuperación del conocimiento.**

Figura 14. Ejemplo de modelo de anotaciones desarrollado para un software de diseño de contenido digital llamado MONO.



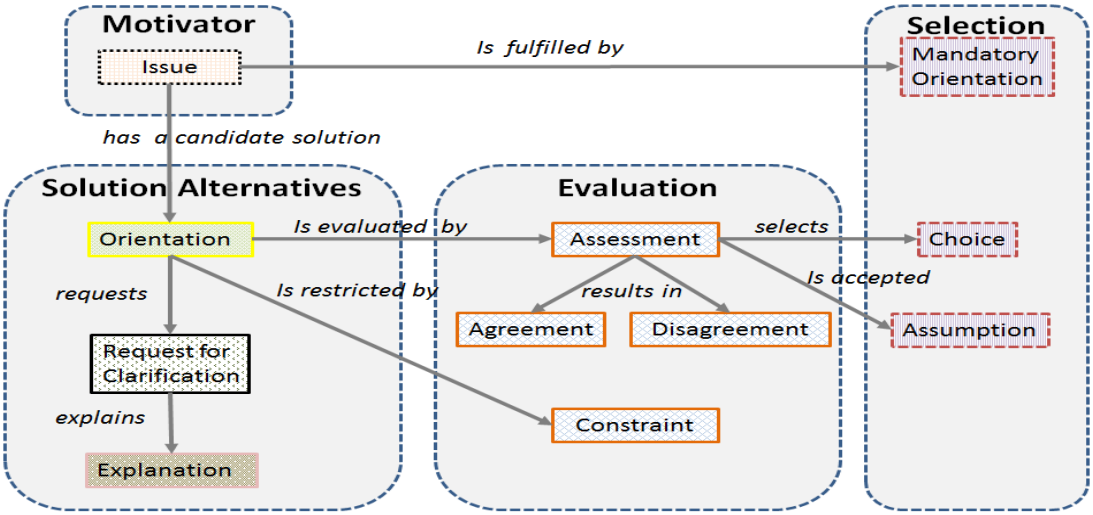
Fuente: PEDRAZA-GARCIA, Gilberto, Correal Darío, Oscar González Rojas, Guillermo Beltrán. MONO+KM: Knowledge Management in Collaborative Project Development:

MONO+KM: Administración de Conocimiento en el Manejo de Proyectos Colaborativos, página 8. 2015.

3.3.3 Representación de las anotaciones como decisiones. Al diseñar el modelo de anotaciones, por cada anotación diseñada en el modelo se debe diseñar su respectivo modelo de decisiones, para así comenzar a tomar decisiones con respecto a cómo vamos a diseñar y solucionar aquella necesidad que surgió.

Si requiere saber cómo diseñar un modelo de decisión, remítase al capítulo 2 en el numeral **Administración del conocimiento.**

Figura 15. Ejemplo de cómo diseñar un modelo de anotaciones a partir de la información recolectada en el catálogo de información.



Fuente: PEDRAZA-GARCIA, Gilberto, Astudillo, Hernán, Correal Darío. DVIA: Understanding how software architects make decisions in design meetings. In Proceedings of the 2015 European Conference on Software Architecture Workshops (ECSAW '15). ACM, New York, NY, USA , Article 51 , 4 pages. 2015.

3.3.4 Wiki Semántica – Semantic MediaWiki (SMW). Se hace uso de la Wiki semántica debido a sus características especiales y ya que es una buena herramienta para compartir los resultados que se lograrán tras la investigación, por lo cual se espera realizar la inclusión de los varios modelos de decisión que tengan como objetivo el aprendizaje de la herramienta, para ellos se hará uso de páginas especial, recurso ya incluido dentro de la Wiki semántica y adicional se utilizará el lenguaje de consultas especializado para Wikis SPARQL.
Para la construcción de la Wiki se espera tener las siguientes características visuales:

- Representaciones de cuadriculas donde incluye información puntual del tema u objeto que se esté describiendo actualmente.

Figura 16. Encabezado de wiki semántica.

name	Definición del estilo de arquitectura.
justification	Se selecciona la alternativa 3 invocación implicita de componentes con resposabilidades independientes porque ofrece mejores ventajas en procesamiento de un gran volumen de información aunque la latencia para los usuarios fnales es inferior
state	Aprobado
names	

Fuente: wiki semántica, Archidecisions[12].

.

- Cuadros de información que contienen mayor detalle de datos en cuanto a un tema puntual se refiere.

Figura 17. Tabla de definición donde definimos los principales temas a tratar dentro de la wiki.

Deci:ArchitecturalStyleDefiniton	
name	Blackboard, un componente simple encargado de procesar y almacenar la información recibida del vehículo. También se encarga del envío de la información por internet
description	Un componente que encapsule toda la funcionalidad para garantizar mejor desempeño y menor cantidad de recursos
source	Shafer86
implication	Alto acoplamiento
names	

Fuente: wiki semántica, Archidecisions[13].

- Links alternativos que direccionan al usuario a un sitio donde la información se complementa de mejor manera según lo que este mismo desee conocer.

Figura 18. Alternativas representadas en una wiki.

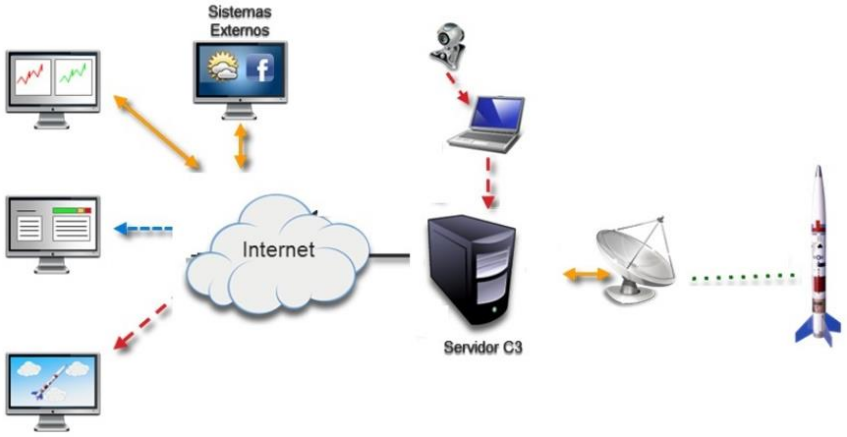
- Architectural Style Alternative 1
- Architectural Style Alternative 2
- Architectural Style Alternative 3

Fuente: wiki semántica, Archidecisions[14]..

[12] PEDRAZA-GARCIA, Gilberto. Wiki con documentación proyecto PUA- Uniandes [marzo de 2015]. Colombia <archidecisions/referata.com >. [Citado Marzo 13 de 2017].
[13] PEDRAZA-GARCIA, Gilberto, referata, Óp. cit.
[14] PEDRAZA-GARCIA, Gilberto, referata, Óp. cit.

- Imágenes que ayuden a la comprensión de lo explicado o que se está apunto de explicar con el fin de que la información tenga mayor facilidad de compresión.

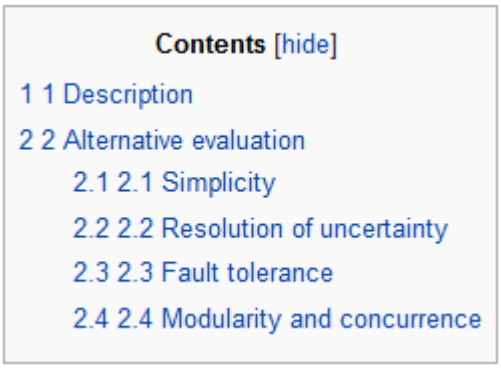
Figura 19. Diagrama representativo de una alternativa.



Fuente: wiki semántica, Archidecisions[16].

- Tablas de contenido que proporcionan al usuario una navegación más cómoda dentro del sitio ubicándolo directamente en el lugar o lugares donde se encuentra la información de mayor interés.

Figura 20. Tabla de contenido de wiki semántica donde se distribuyen los principales temas a exponer.8



Fuente: wiki semántica, Archidecisions[17].

Finalmente se complementará cada ítem del sitio creado en general con la información correspondiente a la investigación haciendo como énfasis al resultado obtenido en esta como lo serán diagramas de decisión, anotaciones encontradas, información de reingeniería del código etc.

[13] PEDRAZA-GARCIA, Gilberto, referata, Óp. cit.
[16] PEDRAZA-GARCIA, Gilberto, referata, Óp. cit.
[17] PEDRAZA-GARCIA, Gilberto, referata, Óp. cit.

4. DESARROLLO METODOLÓGICO

4.1 CLASIFICACIÓN DE INFORMACIÓN DISPONIBLE DE MIFOS X

La información y documentación que se logró extraer de la herramienta MIFOS X se divide en varias secciones por lo cual se dispone a hacer una separación de esta misma dando a conocer la información de manera secuencial iniciando por la información básica hasta la información de mayor relevancia de la herramienta generando de esta manera una clasificación cuyo objetivo es hacer más entendible la investigación realizada.

4.1.1 Documentación básica de MIFOS X. La documentación básica de la herramienta es fundamental para el inicio de un buen conocimiento y desarrollo de futuras mejoras, ya que partir del hecho de conocer lo básico de la herramienta genera un aporte de vital importancia para el usuario cuyo objetivo sea conocer a fondo la funcionalidad al detalle de MIFOS X.

Entre la documentación se encuentra: guía de instalación, manual de usuario, especificaciones funcionales y kit de herramientas para desarrolladores, con esta documentación el usuario se pretende dar a conocer al usuario la herramienta a niveles funcionales y técnicos si así lo desea, a continuación, se describe cada uno de los documentos mencionados anteriormente:

En la guía de instalación se indica el paso a paso de cómo se debe instalar la versión ejecutable de MIFOS X de manera correcta.

- **Instalación de java.** Corresponde a la documentación del paso a paso de la instalación de cada uno de los componentes que deberá llevar la herramienta, por lo cual se divide en varios segmentos entre los cuales hace referencia a:
  Instalación de la máquina virtual de Java (JDK).
  Instalación de MYSQL.
  Instalación del TomCat.
  Creación de variables de entorno.
  Instalación de la herramienta MIFOS X.
  Inicialización de datos en MYSQL con HeidiSQL.
  Configuración del servidor de aplicaciones TomCat.
  Conectar la herramienta al servidor de aplicaciones.
  Como ejecutar la herramienta MIFOS X.

- **Manual de usuario.** El manual de usuario de la herramienta MIFOS X tiene el fin de proporcionar información acerca del uso de esta misma para usuarios que han instalado la aplicación por primera vez y desean

conocer más sobre su funcionalidad generando de esta manera un manual que está compuesto por tres guías que cubren tres diferentes roles dentro de la herramienta que hacen referencia a:
Guía para todos los usuarios: Información sobre el registro y la salida, la navegación y la generación de informes en Mifos X.
Guía para administradores (plataforma MIFOS X). Información para los administradores de Mifos X sobre cómo configurar y configurar Mifos X para las necesidades específicas de su organización.
Guía para operadores (aplicación de la comunidad MIFOS X). Información para el uso diario para crear y administrar entradas de cuenta, transacciones y productos de cuenta en Mifos X.

- **Especificaciones funcionales.** Esta guía explica que una especificación funcional es un documento formal utilizado para describir en detalle para los desarrolladores de software las capacidades, apariencia e interacciones de un producto con los usuarios. La especificación funcional es un tipo de guía y punto de referencia continuo a medida que los desarrolladores escriben el código de programación.

  Antes de que el producto existiera, escribieron la guía del usuario para un sistema de procesamiento de textos, luego declararon que la guía del usuario era la especificación funcional. Por lo general, la especificación funcional para un programa de aplicación con una serie de ventanas interactivas y diálogos con un usuario mostraría el aspecto visual de la interfaz de usuario y describiría cada una de las posibles acciones de entrada de usuario Y las acciones de respuesta del programa. Una especificación funcional también puede contener descripciones formales de las tareas del usuario, dependencias de otros productos y criterios de usabilidad.

  Muchas compañías tienen una guía para desarrolladores que describe qué temas deben incluir las especificaciones funcionales de cualquier producto.

  Para una idea de dónde encaja la especificación funcional en el proceso de desarrollo, aquí hay una serie típica de pasos en el desarrollo de un producto de software.

- **Requisitos**. Esta es una declaración formal de lo que los planificadores de productos informados por su conocimiento del mercado y la entrada específica de clientes existentes o potenciales creen que es necesario para un nuevo producto o una nueva versión de un producto existente. Los requisitos se suelen expresar en términos de declaraciones narrativas y de manera relativamente general.

- **Objetivos**. Los objetivos son escritos por los diseñadores del producto en respuesta a los Requisitos. Describen de una manera más específica cómo será el producto. Los objetivos pueden describir arquitecturas, protocolos y estándares a los cuales el producto se ajustará. Los objetivos medibles son aquellos que establecen algunos criterios por los cuales el producto final puede ser juzgado. La mensurabilidad puede ser en términos de algún índice de satisfacción del cliente o en términos de capacidades y tiempos de tareas. Los objetivos deben reconocer las limitaciones de tiempo y recursos. El programa de desarrollo es a menudo parte o un corolario de los Objetivos.

- **Especificación funcional.** La especificación funcional (por lo general especificaciones funcionales o justspec para abreviar) es la respuesta formal a los objetivos. Describe todas las interfaces externas de usuario y programación que el producto debe soportar.

- **Solicitudes de cambio de diseño.** A lo largo del proceso de desarrollo, cuando se reconoce la necesidad de cambio en la especificación funcional, se describe un cambio formal en una solicitud de cambio de diseño.

- **Especificación lógica.** La estructura de la programación (por ejemplo, grupos principales de módulos de código que soportan una función similar), módulos de código individuales y sus relaciones, y los parámetros de datos que pasan entre sí se pueden describir en un documento formal denominado especificación lógica. La especificación lógica describe las interfaces internas y es para uso exclusivo de los desarrolladores, probadores y, posteriormente, hasta cierto punto, los programadores que prestan servicio al producto y proporcionan correcciones de código al campo.

- **Documentación del usuario.** En general, todos los documentos anteriores (excepto la especificación lógica) se utilizan como material de origen para los manuales técnicos y la información en línea (como páginas de ayuda) preparadas para los usuarios del producto.

  Plan de prueba. La mayoría de los grupos de desarrollo tienen un plan de prueba formal que describe los casos de prueba que ejercerán la programación que se escribe. Las pruebas se realizan a nivel de módulo (o unidad), a nivel de componentes y al nivel del sistema en el contexto de otros productos. Esto puede ser pensado como prueba alfa. El plan también puede permitir la prueba forbeta. Algunas compañías ofrecen

una versión temprana del producto a un grupo seleccionado de clientes para realizar pruebas en una situación de "mundo real".

Idealmente, el producto final es una implementación completa de la especificación funcional y solicitudes de cambio de diseño, algunas de las cuales pueden resultar de pruebas formales y pruebas beta.

- **Kit de herramientas para desarrolladores.** Esta guía fue diseñada y dirigida a aquellas personas que desean ser parte de la comunidad de Desarrolladores MIFOS, en esta se encontrará las siguientes características.

  Código fuentes para comenzar el desarrollo de nuestros propios módulos y funcionalidades.
  Comunidad MIFOS X, la cual ha realizado múltiples aportes a los creadores para implementar y mejorar la Herramienta.
  La documentación de la herramienta en general, diagramas, casos de uso, explicación de módulos y sus clases involucradas.
  Guía instalación y adaptación del código a IDE de desarrollo más conocido para desarrollar JAVA, Eclipse IDE.

4.1.2 Versiones de la herramienta. La herramienta MIFOS X desde su creación ha tenido varias modificaciones las cuales se documentan como nuevas versiones de la herramienta, la primera versión disponible de la herramienta fue a finales del 2013, en la cual la fundación Grameen**,** desarrolló una iniciativa que rompería los paradigmas que existían entre las grandes potencias y los países del tercer mundo, donde las grandes masas en su mayoría de escasos recursos, no disponían de acceso al sistema financiero debido a que no aplicaban a los altos estándares de calificación para acceder a los diferentes productos ofrecidos por estas entidades.

La iniciativa **MIFOS** fue desarrollada principalmente para llegar a todas aquellas poblaciones, de diferentes clases culturales y sociales, para ofrecerles los diferentes productos y servicios con condiciones de prestaciones de servicios muy flexibles y un buen buffet de servicios al alcance de todos.

En cuanto a la herramienta MIFOS X desarrollada para apoyar la iniciativa MIFOS a continuación se presentará las fechas exactas de cada uno de sus versiones hasta la versión que se dispone hoy día.

Figura 21. Todas las versiones que se han generado a partir, de su implementación.

Version	Code name	Release date	Supported until
16.03.3	NA	2016-3-23	**2016-03-23**
16.01.2	NA	2016-1-21	2016-01-21
16.01.1	NA	2016-1-13	2016-01-13
15.12.2	NA	2015-1-17	2015-12-17
15.11.2	NA	2015-18-11	2015-9-11
15.11.1	NA	2015-9-11	2015-9-11
15.10.2	NA	2015-21.10	2015-10-21
15.09.3	NA	2015-09-16	2015-09-16
15.03.1	NA	2015-05-08	2015-05-08
15.03	NA	2015-04-06	2015-05-01
1.26.0	NA	2014-12-23	2015-02-01
1.25.1	NA	2014-11-03	2014-12-01
1.24.0	NA	2014-07-10	2014-08-01
1.23.1	NA	2014-06-18	2014-07-01
1.23	NA	2014-06-17	2014-07-01
1.22	NA	2014-05-01	2014-06-18
Old version	Older version, still supported	**Latest version**	Future release

Fuente: Wikipedia MIFOS, repositorios

Cuando la primera versión de la herramienta salió al mercado, esta genero un gran impacto entre las diferentes comunidades de software libre, cada versión data de cómo fue que evoluciono esta herramienta, con sus distintos módulos financieros, lenguajes de programación y novedosas mejoras que se ven reflejadas en cada versión como se muestra a continuación:

Figura 22. Todas las versiones fueron almacenadas en servidores virtualizados de Amazon EC2 AMI, los cuales se encuentran disponibles, si se tiene una cuenta.

Zone	Name	version	Arch	Instance Type	Release	AMI-ID	AKI-ID
ap-southeast-1	Mifos X 16.03.4	16.03.3	x86_64	ebs	2016-03-23	ami-a875a0cb	aki-503e7402
ap-southeast-1	Mifos Platform 16.01.2	16.01.2	x86_64	ebs	2016-01-21	ami-56468a35	aki-503e7402
ap-southeast-1	Mifos Platform 16.01.1	16.01.1	x86_64	ebs	2016-01-12	ami-57814d34	aki-503e7402
ap-southeast-1	Mifos Platform 15.12.2	15.12.2	x86_64	ebs	2015-11-18	ami-e629eb85	aki-503e7402
ap-southeast-1	Mifos Platform 15.11.2	15.11.2	x86_64	ebs	2015-11-18	ami-f1915092	aki-503e7402
ap-southeast-1	Mifos Platform 15.11.1	15.11.1	x86_64	ebs	2015-11-9	ami-6fa96f0c	aki-503e7402
ap-southeast-1	Mifos Platform 15.10.2	15.10.2	x86_64	ebs	2015-09-16	ami-dc6c7e8e	aki-503e7402
ap-southeast-1	Mifos Platform 15.09.3	15.09.3	x86_64	ebs	2015-09-16	ami-acfbeefe	aki-503e7402
ap-southeast-1	Mifos Platform 15.03.1	15.03.1	x86_64	ebs	2015-05-08	ami-2a734b78	aki-503e7402
ap-southeast-1	Mifos Platform 15.03	15.03	x86_64	ebs	2015-04-06	ami-145d6046	aki-503e7402
ap-southeast-1	Mifos Platform 1.26.0	1.26.0	x86_64	ebs	2014-12-23	ami-909bb3c2	aki-503e7402
ap-southeast-1	Mifos Platform 1.25.1	1.25.1	x86_64	ebs	2014-11-03	ami-ab577af9	aki-503e7402
ap-southeast-1	Mifos Platform 1.24.0	1.24.0	x86_64	ebs	2014-07-11	ami-a6035af4	aki-503e7402
ap-southeast-1	Mifos Platform 1.23.1	1.23.1	x86_64	ebs	2014-06-18	ami-8ace91d8	aki-503e7402
ap-southeast-1	Mifos Platform 1.22.0	1.22	x86_64	ebs	2014-05-01	ami-50693b02	aki-71665e05

Fuente: Wikipedia MIFOS, repositorios

A continuación, se presentará la información disponible de las versiones más significativas por las cuales paso la herramienta MIFOS X.

Tabla 5. características de cada versión de MIFOS, a partir de la primera entrega de la herramienta en el 2013, hasta el 2016.

Versión	Año De Publicación	Descripción	Características
• **MIFOS X** 2.6.3	• Septiembre 26 del 2013	**MIFOS X** Representa la próxima generación de MIFOS, es la plataforma de tecnología abierta para la inclusión financiera de las personas de bajos recursos, Su objetivo es cumplir con nuestra visión a largo plazo de una plataforma completamente extensible que puede escalar a través de multi-alquiler y rápidamente extenderse a través de una arquitectura en capas limpia y completa API. . Todos estos servicios están expuestos a través de	• Administración de **clientes**: Hemos hecho la gestión de clientes mucho más robusta, con la posibilidad de cargar documentos y realizar seguimiento de los documentos de identidad de los clientes. • **Gestión de la cartera:** la creación de productos de préstamos es aún más flexible, ahora puede verse de forma dinámica los productos de crédito que se crean para su previa administración. • **Contabilidad:** Hemos puesto un módulo de contabilidad de caja integrada con seguimiento de cartera automatizada, así como la capacidad de entrada manual de modificaciones de cartera. • **Gestión de usuarios:** Encontrará el mismo módulo para conceder permisos a las diferentes bases de datos, pero ahora con el

		una API REST para crear rápidamente nuevas aplicaciones de cliente front-end para los servicios financieros a los pobres.	concepto corrector de ortografía para que pueda garantizar una mejor integridad de los datos. • **Informes:** Continu amos utilizando Pentaho como nuestro motor de informes, pero también hemos añadido otra capa de informes basados en pantallas de peso ligero para un total de 30 informes, incluyendo informes financieros estándar.
MIFOS X 1.16.0	15 de febrero del 2014	Para que los usuarios, MIFOS X le ofrece una plataforma en la que se puede implementar varias aplicaciones y módulos de apoyo a cualquier modelo que se utiliza para ofrecer servicios financieros a los pobres. MIFOS X es ligero, fácil de usar, barato para alojar y listo para extender.	• **Rapidez:** la rapidez con la que la aplicación se ejecuta desde un servidor, es muy importante para soportar las operaciones en los distintos servicios que presta esta plataforma. • **Implementación Créditos de grupo:** Este módulo se implementó a petición de los distintos usuarios y clientes de la plataforma, esta modalidad esta soportada por múltiples plataformas, tales como **Spring**

			framework, jersey, Spring Security, Spring JDBC, java 1.6, Tomcat 7, MySQL 5.1.
MIFOS X 1.17.3	Junio 25 del 2015	MIFOS X se pueden implementar en cualquier entorno: nube o instalado, o fuera de línea, móvil o PC; es lo suficientemente extensible para soportar cualquier tipo de organización o canal de distribución, y suficiente para mantener a cualquier producto, servicio o metodología flexible.	**Origen del control y de presentación Código** • **GITHUB:** Nosotros usamos Git , un sistema distribuido de control de versiones de código abierto para ayudarnos y colaborar de manera eficiente A comunicar los cambios al momento de escribir código. • **GIT Y SU FLUJO DE TRABAJO:** Mifos X en GitHub sirve como un repositorio centralizado. Todos los contribuyentes deben Descargar este repositorio en lugar de hacer directamente una solicitud de extracción. Hay dos ramas principales, *maestro* y *desarrollan* con desarrollador se genera la integración. • **Arquitectura:** MIFOS X posee una arquitectura diseñada en Java que contiene la actividades principales más

			complejas y funcionalidad técnica para la inclusión financiera. • **Integración Continua:** Travis CI es un servicio de integración continua alojada que se integra muy bien con GitHub. La usamos tanto para nuestra plataforma de MIFOS X y la aplicación de la comunidad para asegurar que con cada cambio de código.

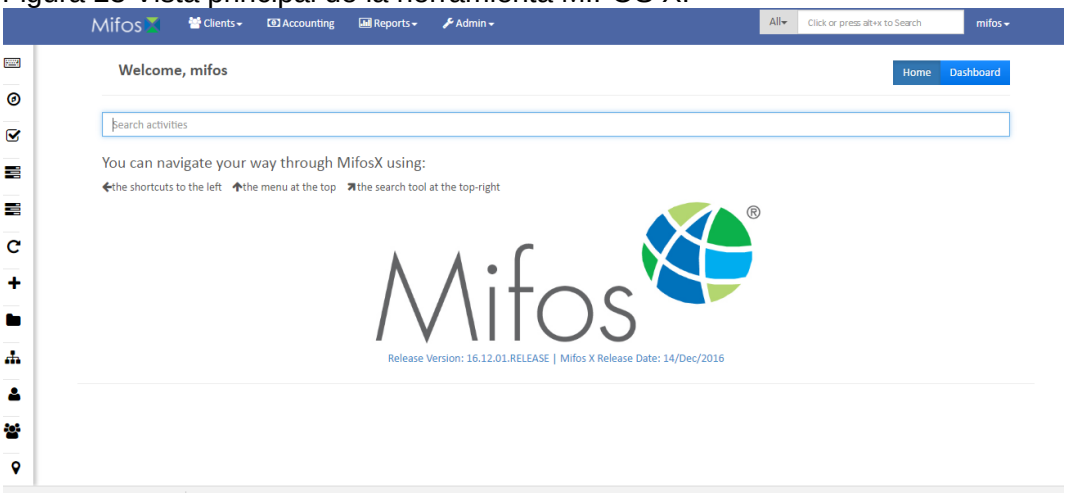
Fuente: transcendencia MIFOS. www.archivosweb.com

4.1.3 Código fuente. El código fuente de la aplicación se encuentra divido en dos partes:

- **FRONT-END**. encargado es la parte del desarrollo web que se dedica de la parte frontal de la herramienta, es decir, del diseño del sitio, desde su estructura hasta los diferentes estilos como colores, fondos, tamaños, tipo de letra, etc. Hasta llegar a las animaciones y efectos.

  A continuación, se aprecia el FRONT-END de la herramienta MIFOS X:

Figura 23 Vista principal de la herramienta MIFOS X.

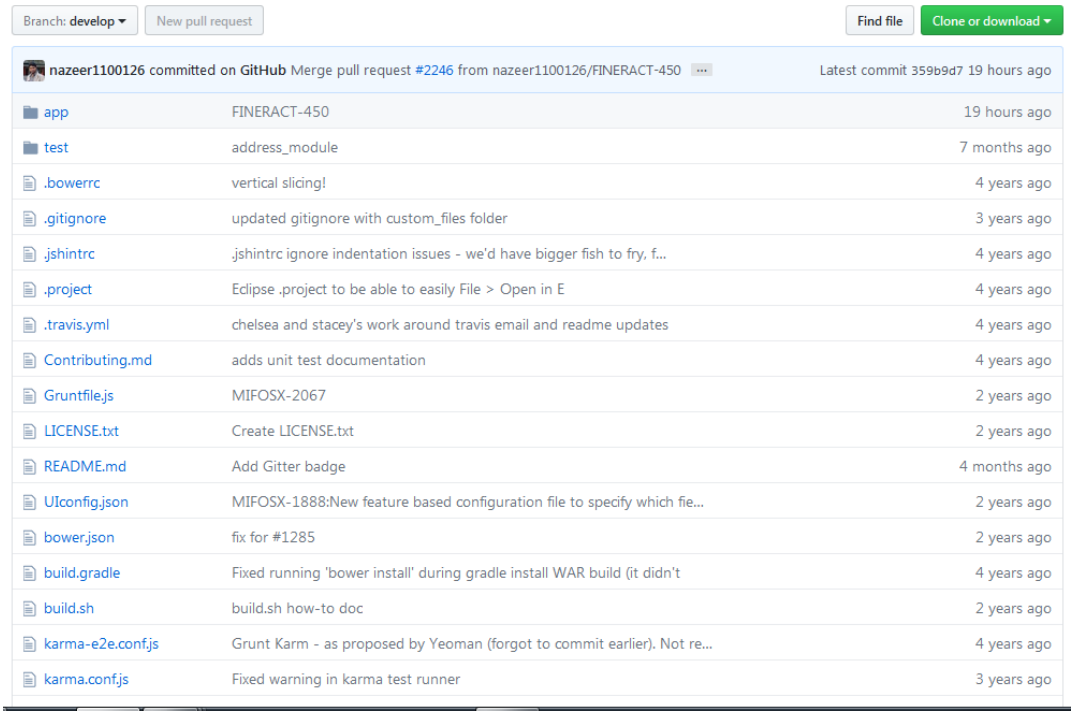


Fuente: página principal de MIFOS plataforma web.

El código fuente del FRONT-END es llamado "*Community-app*" el cual se encuentra almacenado dentro de un repositorio github con el fin de que sea accesible para todos los usuarios, cabe destacar que la herramienta está escrita en tecnologías web estándar como JavaScript, CSS y HTML5, aprovechando bibliotecas como AngularJS, Bootstrap y Font Awesome.

A continuación, se presenta el repositorio donde está almacenado el código fuente del FRONT-END[18].

Figura 24. Repositorio donde se almacena el código fuente del FRONT-END.



Fuente: repositorio de código guardado en MIFOS repositorio GITHUB

Como se puede observar se cuenta con todos los componentes que integran la parte FRONT del código, así como la posibilidad de clonar este repositorio en un IDE de desarrollo con el fin de manipular este a necesidad.

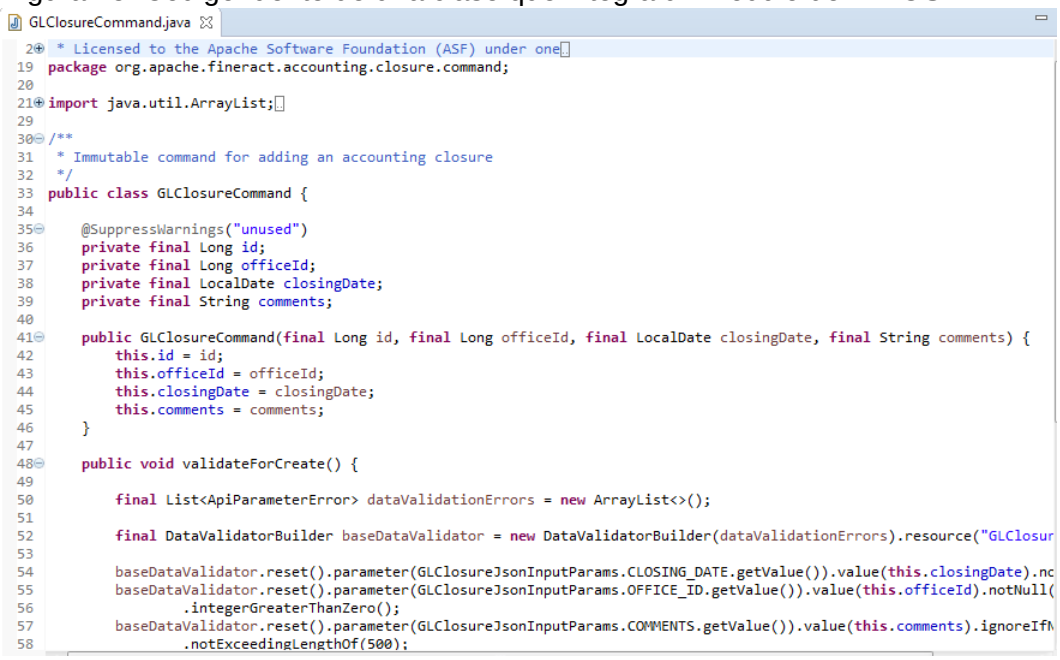
- **BACK-END.** el lado opuesto del FRONT-END, es el encargado de **la parte lógica de la herramienta, también encargado de que todo funcione como correctamente**, el BACK-END es la parte de atrás que de alguna manera no es visible para el usuario ya que no se trata de diseño, o elementos

[18] REPOSITORIO FRONT. from end platform MIFOS X. Estados Unidos. <https://github.com/openMF/community-app#mifosx-community-app>. [Citado septiembre 1 de 2016]

gráficos, se trata de programar las funciones que tendrá la herramienta. El BACK-END es la programación dura y pura, desde la programación de las funciones de la herramienta hasta bases de datos e incluso más[19].

A continuación, se presenta parte del código fuente del BACK-END el cual está desarrollado en Java:

Figura 25. Código fuente de una clase que integra un módulo de MIFOS X.



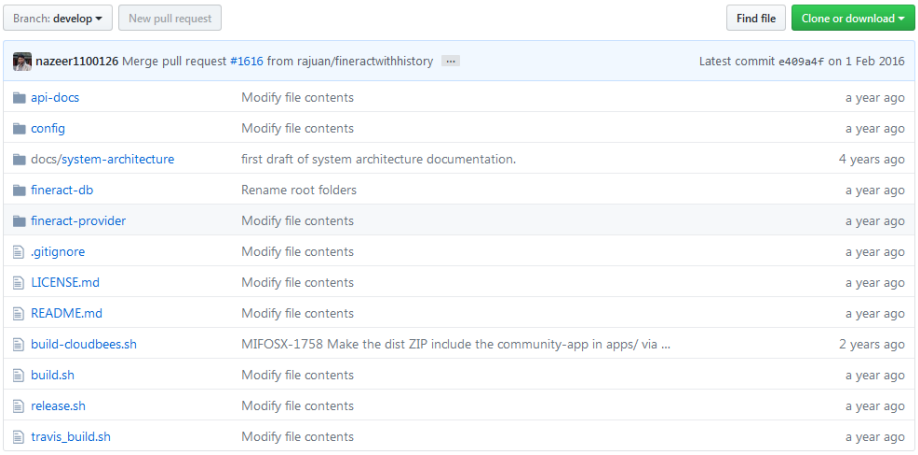
Fuente: Autores

Al igual que el FRONT-END, el código fuente del BACK-END esta almacenado en un reposito con la misma finalidad que el anterior, como ya se mencionó anteriormente este está desarrollado en Java y adicional tiene las respectivas conexiones a bases de datos, el BACK-END es llamado "*incubator-fineract*".

A continuación, se presenta el repositorio donde se encuentra almacenado el código fuente del BACK-END:

Figura 26. Repositorio donde se almacena el código fuente del BACK-END.

[19] REPOSITORIO BACK. back end platform MIFOS X. Estados Unidos. < https://github.com/openMF/community-app#mifosx-community-app.>. [Citado septiembre 1 de 2016]

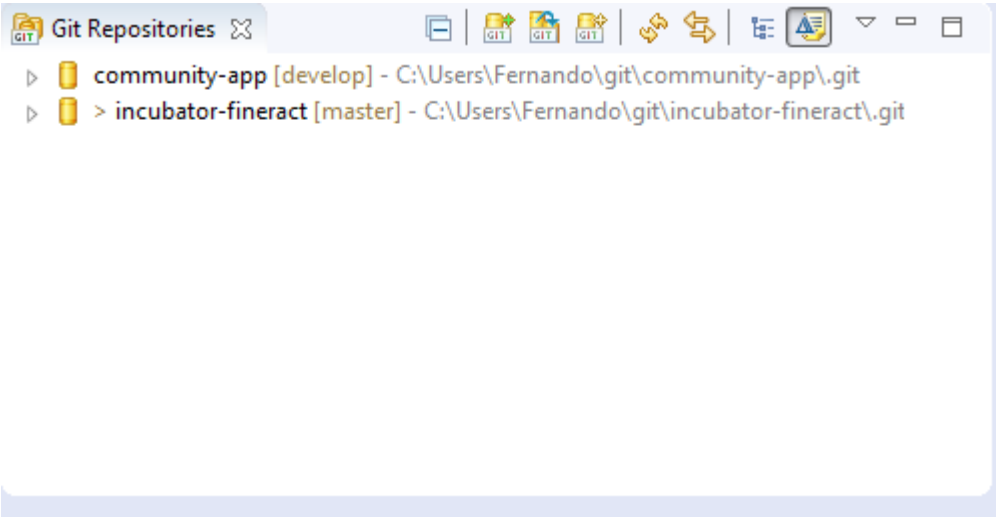


Fuente: repositorio almacena en gibhub.

Como se puede observar en el repositorio del BACK-END se cuenta con todos los componentes necesarios y su respectiva opción para la clonación del repositorio y así poder manipular su código a necesidad.

En la siguiente imagen se presenta la clonación de ambos repositorios en un entorno de desarrollo (en este caso Eclipse) para la libre manipulación de la herramienta en general según las necesidades que se lleguen a presentar:

Figura 27. Repositorios almacenados en IDE de desarrollo Eclipse.



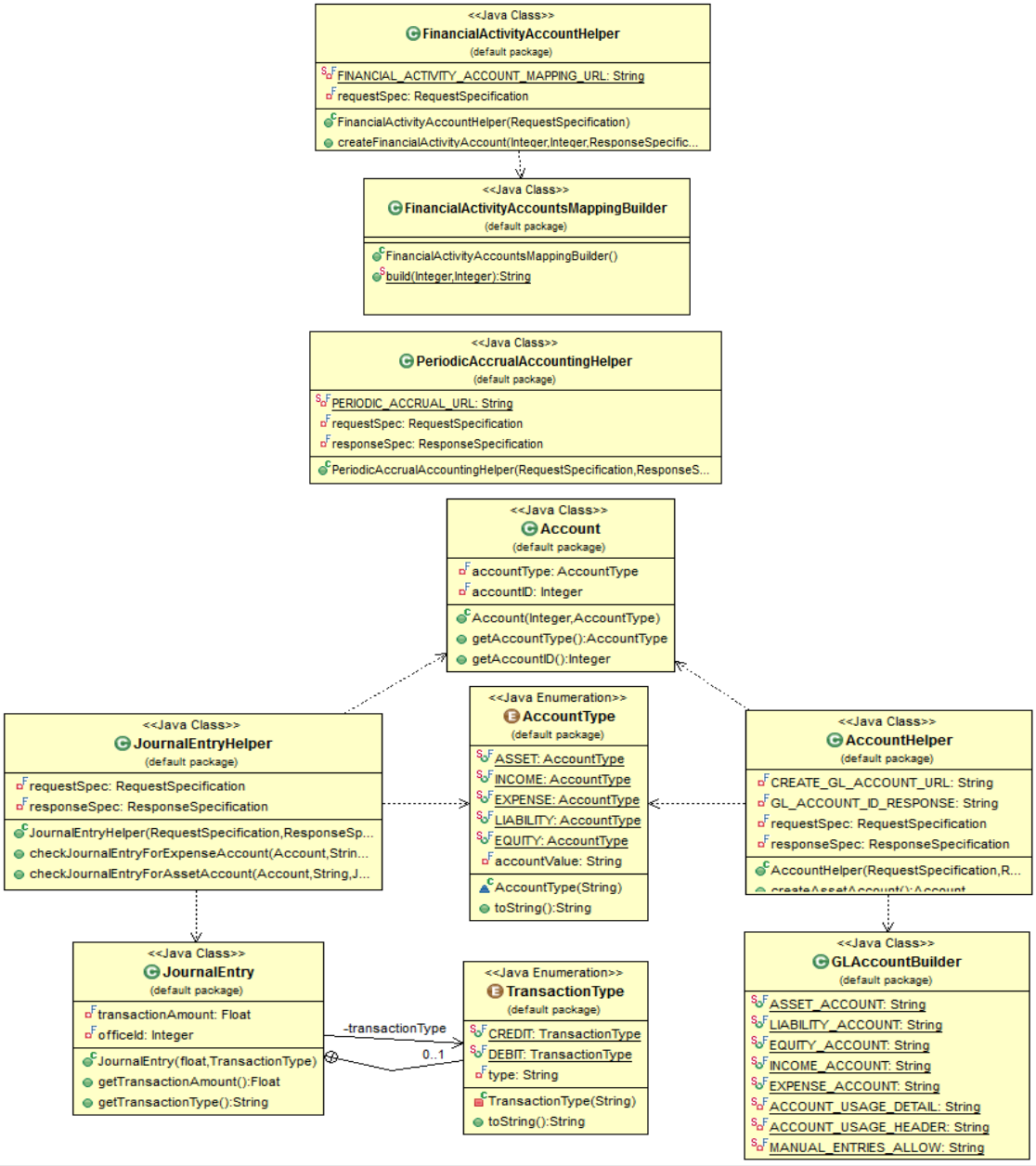
Fuente: Autores

Estos repositorios son de vital importancia para el conocimiento interno de la herramienta, ya que el FRONT-END como se mencionó anteriormente es la parte visual, es importante mencionar que el proceso de ingeniería inversa de la herramienta será destinado a la parte lógica de esta, es decir, se enfocará en el BACK-END.

Tener a mano el código fuente el cual será sometido a una herramienta de ingeniería inversa presenta una fuente vital de información en especial si este es enfocado a la parte lógica (BACK-END).

4.1.4 Utilizando la herramienta en el BACK-END de MIFOS X. Utilizando la herramienta ObjectAid UML en el código de MIFOS X más exactamente en el módulo de contabilidad se obtiene el siguiente resultado:

Figura 28. Diagrama UML de clases resultante del módulo garantías.



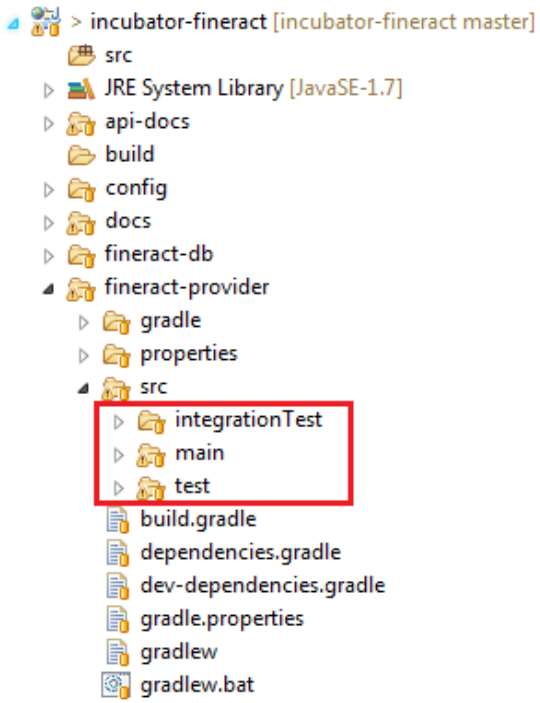
Fuente: Autores

4.1.4.1 Análisis. Como se observa al usar la herramienta en parte del código, este genera diferentes diagramas UML (Unified Modeling Language), lo cual identifica a simple vista las diferentes relaciones que hay entre las diferentes clases, generando de esta manera un conocimiento inicial de las diferentes relaciones que pueden existir entre sus diferentes clases, así como el conocimiento básico que proporciona los diagramas UML como lo son individualmente el nombre de sus respectivas clases, métodos que componen la clase, variables de clase entre otras características básicas que se puede conocer por medio de este método.

Entendiendo que los diagramas anteriores corresponden a un único modulo es importante mencionar que una estructura similar de clases con sus respectivas relaciones existe en el número significativo de módulos que contiene el BACK-END de la aplicación por lo cual adicional a la relación que hay entre clases de cada módulo, existe una cantidad de relaciones puntuales entre cada módulo, generando de esta manera no solo dependencia entre clases sino también dependencia entre clases de diferentes módulos.

A continuación, se muestra la relación de módulos directamente en el repositorio que se descargó e incluyo como proyecto local en el IDE de desarrollo Eclipse:

Figura 29. Módulos principales del BACK-END.

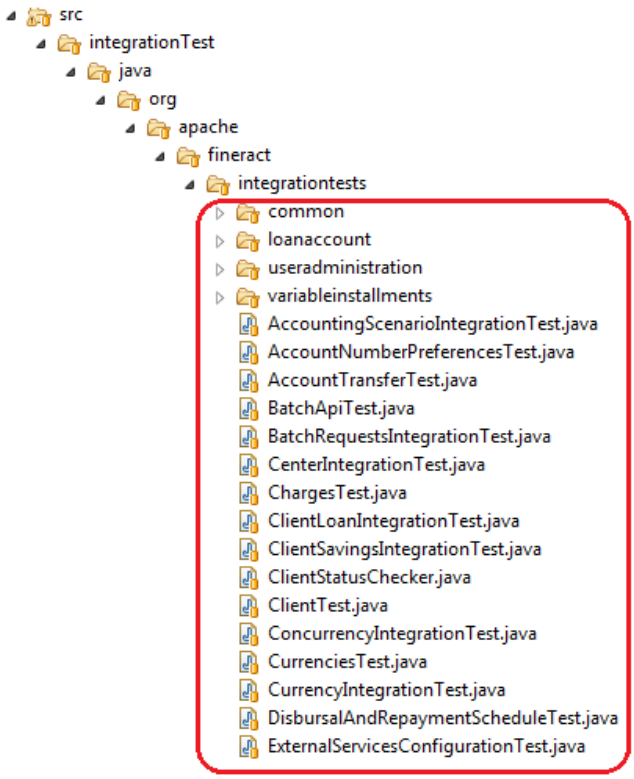


Fuente: Autores

Como se puede observar dentro de la carpeta del BACK-END llamada "*incubator-fineract*" directamente en la carpeta de desarrollo "*src*" esta contiene tres módulos principales "*integrationTest*", "*main*" y "*test*", cada uno de estos módulos tiene sus respectivas funciones, pero para el caso se centrará la atención en el módulo "*integrationTest*" ya que contiene los módulos de las principales funciones de que ejecuta la herramienta como por ejemplo el módulo de contabilidad.

A continuación, se muestra los módulos contenidos dentro del módulo "*integrationTest*" que como ya se mencionó hace referencia a las operaciones principales de la aplicación.

Figura 30. Módulos que integran el módulo principal "IntegrationTest".



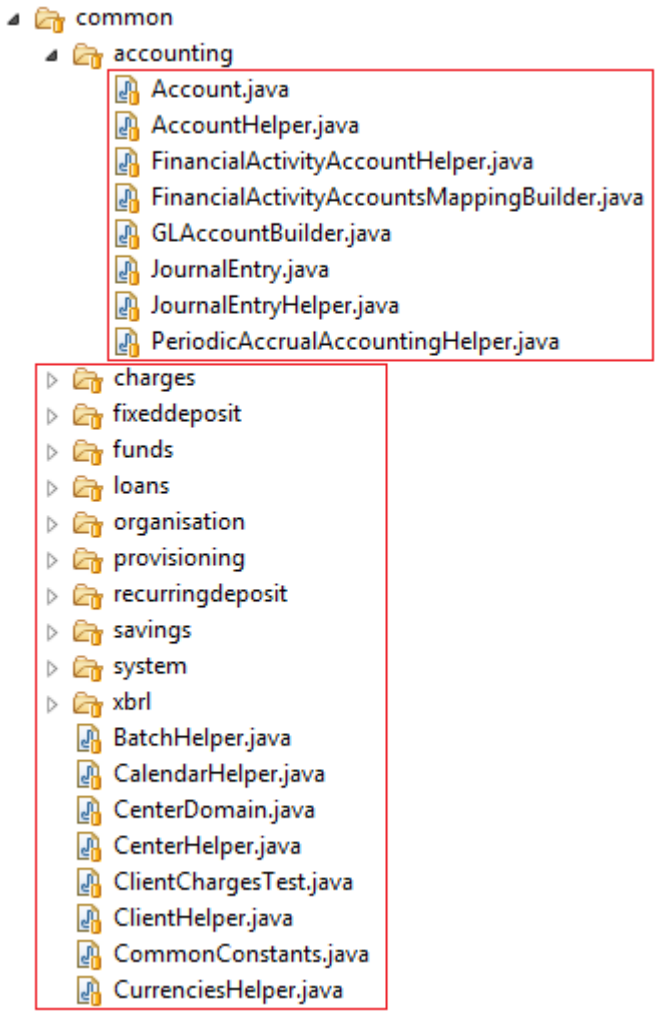
Fuente: Autores

Se identifica que al módulo "*integrationTest*" lo compone cierta cantidad de módulos adicionales, así como clases que no se encuentran almacenadas dentro de modulo pero que pueden o no dependiendo de la temática del módulo tener dependencias con las clases que lo componen, aunque es importante mencionar que también contiene clases independientes, es decir, que no están sujetas a decisión que se cambien en clases de otros módulos, por lo cual representan de ser necesario cambios mínimos en estas.

Al realizar a nivel interno un análisis mayor, a continuación, se muestra el módulo de contabilidad el cual se utilizó de ejemplo en la generación de

diagramas UML, visualizando de esta manera los módulos que comparten relación entre sí y lo cual resulta ser de vital importancia tener en cuenta dichas relaciones a la hora de modificar el código:

Figura 31. Clases que integran el módulo de garantías.



Fuente: Autores

En la primera parte se puede observar el módulo “accounting” utilizado como ejemplo para el uso de la herramienta ObjectAid y las respectivas clases que componen este módulo, seguido se puede observar otros módulos con temáticas diferentes pero que tienen en relación determinados componentes, adicional de otras clases que no se encuentran dentro de esta fase de módulos, esto indica que en caso de realizar modificaciones a alguno de los módulos en cuestión por cambio de necesidades o mejoras es de gran ayuda realizar este mismo análisis en cada uno de los módulos.

Debido a la cantidad de módulos y adicional las clases que componen a cada uno de estos, así como las clases independiente que componen la aplicación, es recomendable realizar el proceso de análisis anterior a módulos por

separado, ya que dependiendo de la herramienta de ingeniería inversa se pueden generar dos inconvenientes:

- La herramienta no soporta la cantidad de módulos y clases contenidas en el proyecto por lo cual no se puede generar información referente a la herramienta.
- Debido a la cantidad masiva de información (módulos y clases) se genera información de reingeniería de esta misma en una escala significativa lo cual dificulta el entendimiento proporcionado por la herramienta.

Debido a estas dos razones principales entre otras se hace más factible realizar este análisis por módulos.

Dentro de la comunidad MIFOS existe un grupo de personas que se dedican al análisis de estos módulos generando de esta manera mejoras o nuevas funcionalidades para la herramienta, el proceso que se sigue es inicialmente encontrar una necesidad, la cual se documentará y se expondrá, generando de esta manera su aprobación o negación por parte de otros miembros de la comunidad, una vez aprobada se empieza a trabajar en la solución más adecuada.

Finalmente, tras plantear una solución se realiza un análisis módulo por módulo de las clases que se verán comprometidas con dicho cambio lo cual llega a generar cambios en la solución esto debido a las decisiones que se deben tomar al momento del desarrollo y tener una visión general del resultado después del cambio punto que se explicará al detalle más adelante.

Es importante mencionar que miembros de la comunidad MIFOS realizan este tipo de análisis con módulo de manera individual generando de esta manera una nueva necesidad o mejorando alguna ya existente, las propuestas de soluciones se basan en un número significativo de decisiones las cuales de las cuales se debe tener el conocimiento general del resultado que conlleva tomar dicha decisión, por lo cual se plantean varios diagramas de decisión con el fin de elegir la mejor solución posible.

4.2 APLICACIÓN DEL MODELO DE ANOTACIONES Y MAPEO DE DECISIONES

La aplicación del modelo de anotaciones, tuvo como principal modelo, tres requerimientos plateados por la comunidad de MIFOS, en ellas se manifestaron una necesidad a partir de lo que los usuarios MIFOS requerían para que la herramienta les diera el potencial necesario para lograr un máximo desempeño y gran utilidad.

A continuación, se describirán y se plasmarán algunos modelos de decisión.

4.2.1 Recuperación de anotación. Modelo de anotaciones, implementación del módulo de garantías propuesto a partir de la necesidad de un usuario.

4.2.1.1 Ruta de recuperación primer modelo de anotaciones. La ruta de recuperación de la información de un modelo de anotaciones, se basa en una investigación exhaustiva, en donde el o motivadores deben especificar al detalle cada uno de elementos, recursos e información, para saber el origen del modelo de anotaciones y así poder realizar un pleno desarrollo del modelo de anotación y sus diferentes componentes.

La ruta está compuesta por los siguientes elementos que se dispusieron para llevar acabo el desarrollo de este modelo, y son los siguientes.

4.2.1.2 MIFOS. Es una iniciativa creada por la fundación Grameen, una empresa dedicada a diseño e implementación de herramientas de software libre, su principal producto una plataforma de micro finanzas capaz de soportar múltiples servicios para el sector de las micro finanzas a nivel mundial, MIFOS X esta plataforma posee una gran acogida en la comunidad de software libre, ya que le permite al usuario personalizar los servicios que desea implementar en su empresa.

Figura 32. Portal oficial de MIFOS X.

Fuente: página oficial MIFOS.

En este portal se encuentra todas las herramientas necesarias para desarrolladores de la herramienta MIFOS X, entre ellos tenemos:

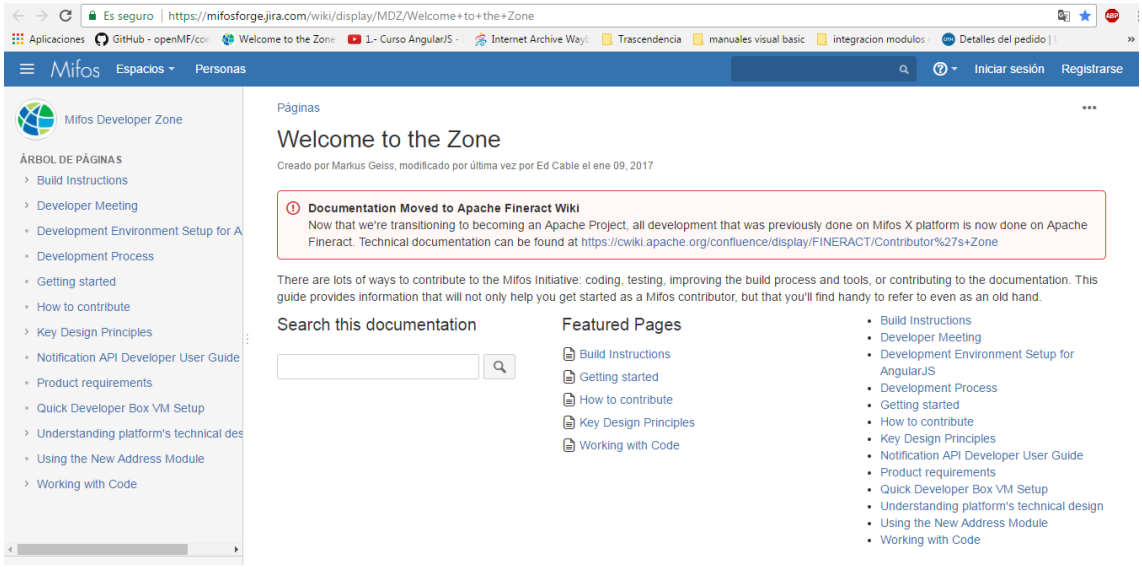
- Especificaciones funcionales.
- Documentación
- Repositorio de código fuente.
- manual de instalación de herramienta para desarrollar en eclipse
- manual de instalación de aplicación base.

- Wiki jira

4.2.1.3 Wiki jira. Es una wiki **diseñada** para informar al usuario sobre todas las funcionalidades, manuales y miembros pertenecientes a la comunidad MIFOS, en donde el usuario tiene múltiples opciones de consulta, entre ellas esta:

- Información usuarios pertenecientes a la comunidad MIFOS.
- Un foro de consulta y desarrollo de nuevas mejoras para MIFOS.
- Un foro de especificaciones funcionales por parte de los usuarios.

Figura 33. Página oficial de JIRA.



Fuente: wiki Jira[20].

4.2.1.4 Jira "Paginas". En este portal encontraremos todos los requerimientos, actualizaciones, sugerencias y soluciones propuestas por la comunidad MIFOS.

Figura 34. Wiki Jira, portal principal de últimas actualizaciones implementadas por los usuarios MIFOS.

[20] WIKI COMUNIDAD MIFOS. Mifos Jira. Estados Unidos. <https://mifosforge.jira.com/secure/Dashboard.jspa>. [Citado febrero 15 de 2017].

Fuente: wiki jira, requerimientos funcionales MIFOS.

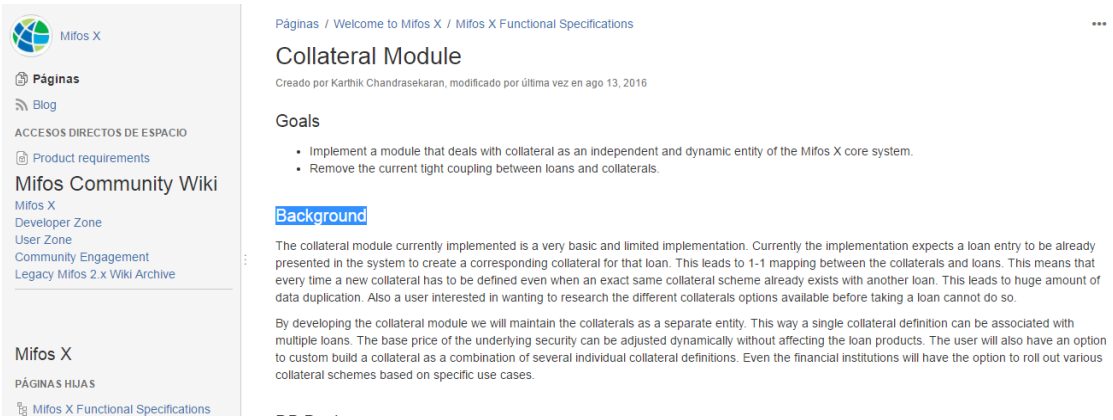
4.2.1.5 Módulo de Garantías. Este es nuestro objeto de estudio, un requerimiento plateado por un usuario el cual manifiesta la necesidad de implementar este módulo dentro de la plataforma MIFOS, en el encontraremos la siguiente información:

- Metas a las que quiere llegar el Usuario que propuso la mejora.
- Detalle del requerimiento sugerido por el Usuario.
- Diagramas de BD, Clase, Requerimientos funcionales.
- Documentos anexos código fuente, archivos planos, contactos.

Figura 35. Localización de primer requerimiento.

Fuente: wiki jira, requerimientos funcionales MIFOS, módulo de garantías.

Figura 36. Wiki Jira, ingreso de últimas actualizaciones de wiki Jira.



Fuente: wiki jira, requerimientos funcionales MIFOS, módulo de garantías primer requerimiento.

La necesidad fue propuesta por un miembro de la comunidad conocido por alias de **Karthik Chandrasekaran**, donde el usuario tras una revisión exhaustiva de la herramienta descubre una posible inconsistencia por lo que el usuario indica lo siguiente.

4.2.1.6 Problema manifestado por el usuario Karthik Chandrasekaran. El módulo de garantía actualmente implementado muy básico y limitado. Actualmente la implementación espera que una entrada de préstamo sea ya presentada en el sistema para crear una garantía correspondiente para ese préstamo. Esto lleva a 1-1 el mapeo entre los requisitos y préstamos. Esto significa que cada vez que se crea un nuevo requisito tiene que ser definido nuevamente, cuando se ingresa un nuevo crédito.

Esto conduce a una enorme cantidad de duplicación de datos. También un usuario interesado en querer investigar las diversas opciones de requisitos disponibles antes de tomar un préstamo se le puede hacer tedioso este proceso.

4.2.1.7 Solución propuesta por el usuario Karthik Chandrasekaran. “Implementar una nueva entidad en la estructura de la base de datos llamada ‘**requisitos_Clientes’** y conectarla con un identificador de código de tipo entero a la tabla **‘Requisitos_Cliente’** así con esta nueva tabla una vez se ingresa el código de crédito de un cliente este estará conectada a todo su historial crediticio evitando así una duplicación innecesaria de datos”.

4.2.1.8 Metas. Las metas comprenden dos factores primordiales:

- Implementar un módulo que se ocupa de las garantías como una entidad independiente y dinámica del sistema central de Mifos X.
- Eliminar el acoplamiento estrecho actual entre préstamos y colaterales.

4.2.1.9 Datos de usuario.

Tabla 6. Datos del primer usuario de la comunidad, para motivar la primera decisión.

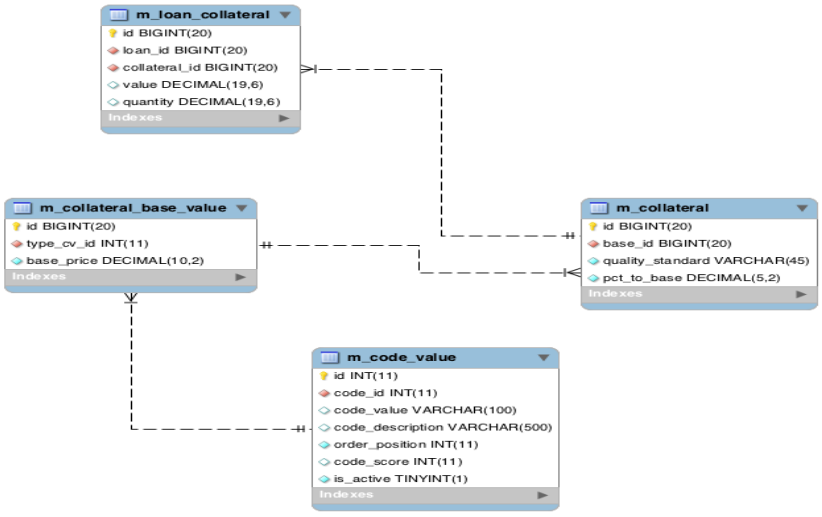
Datos Personales del motivador	Necesidad manifestada	Solución propuesta
Nombre completo: Karthik Chandrasekaran **Correo:** karthikiyer2000@gmail.com **País de origen: india**	Módulo de garantías para evitar replicación de información, cada vez que se generaba un crédito.	Mejorar el módulo de garantías, para evitar replicación de información, cuando se ingresa la información a crediticia del cliente.

Fuente: https://mifosforge.jira.com/wiki/spaces/MIFOSX/pages/149749763/Collateral+Module

A continuación, se mostrará la decisión que se encuentra propuesta en la wiki oficial de la iniciativa MIFOS en la cual tienen participación todos sus usuarios inscritos que tienen procedencia de diferentes partes del mundo.

4.2.2 mapeo de anotación a decisión. Para el modelo anterior se implementará un diagrama de decisiones, con el fin de saber por qué surgió esta necesidad, la cual fue planteada por el usuario **Karthik Chandrasekaran** quien pertenece a la comunidad MIFOS, las principales alternativas con las que cuenta son, la posible evaluación del caso en potencia y por último la decisión final de como satisfacer esta necesidad planteada, para ello se hace uso del siguiente diagrama entidad-relación:

Figura 37. Modelo entidad Relación para implementación de módulo de garantías para préstamos clientes.

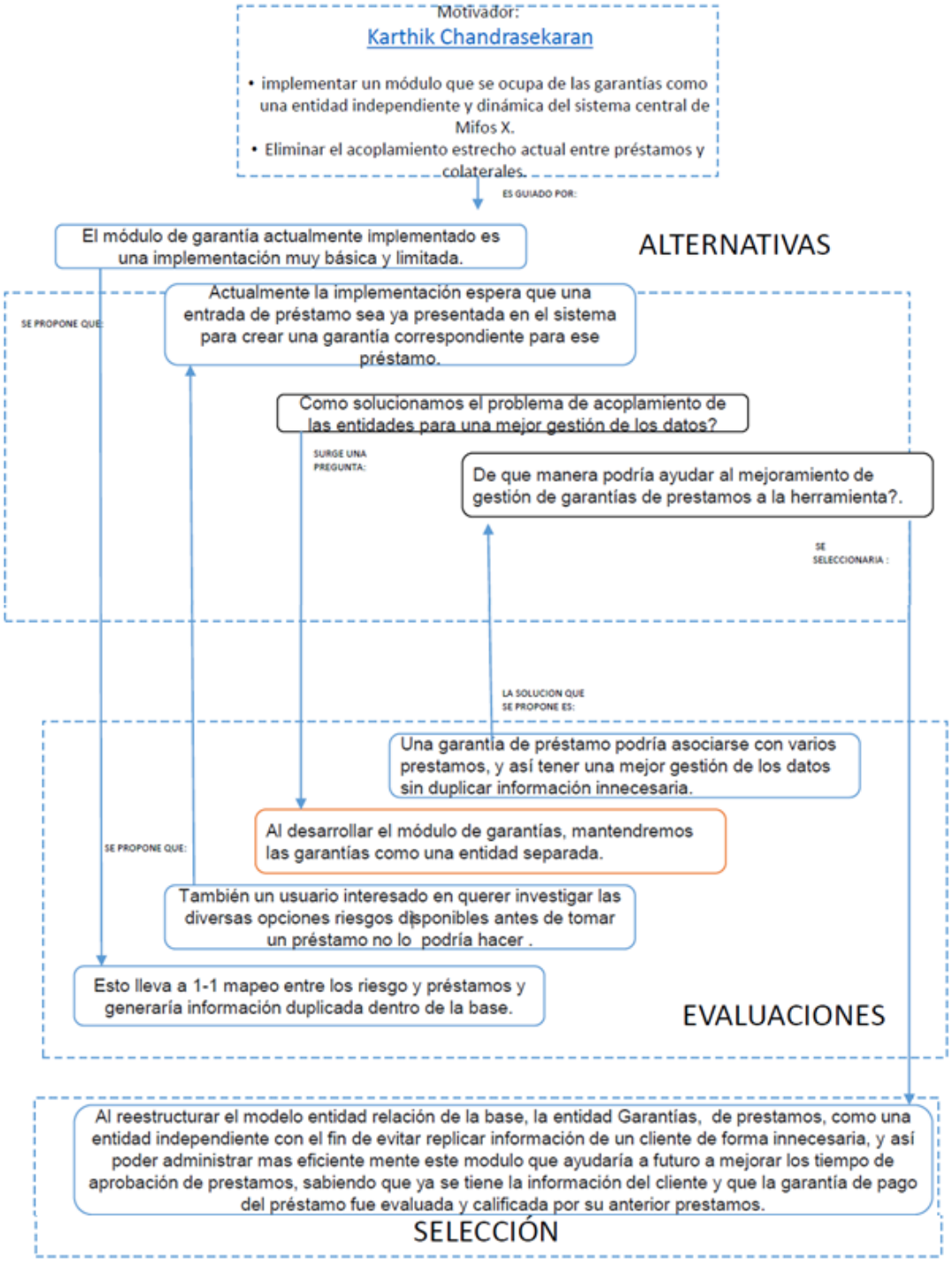


Fuente: wiki Jira, requerimientos funcionales MIFOS.

En cuanto al diagrama de decisión es planteado de la siguiente manera en base a la anotación recuperada previamente:

4.2.3 Modelo de decisiones N° 1 Motivador “Karthik Chandrasekaran”.

Figura 38. Primer modelo de anotaciones, implementación de módulo de garantías.



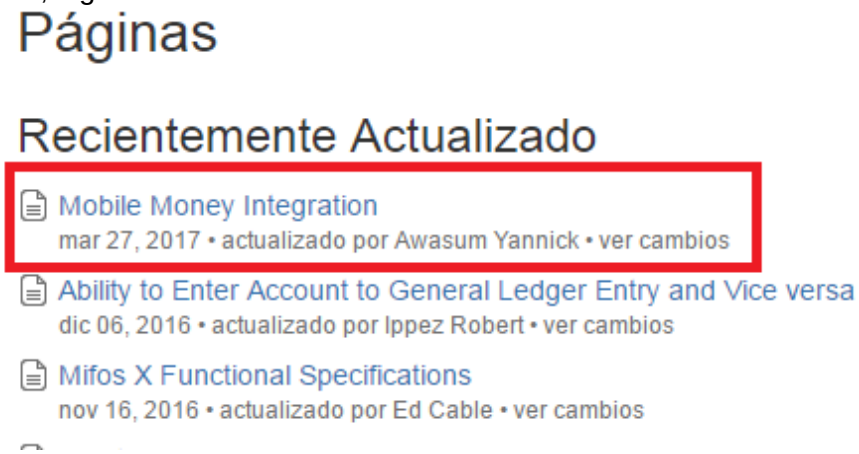
Fuente: Autores.

Modelo de anotaciones, integración de banca móvil propuesto a partir de la necesidad de varios usuarios.

4.2.4 Módulo de integración de banca móvil. Este es nuestro objeto de estudio, un requerimiento plateado por un usuario el cual manifiesta la necesidad de implementar este módulo dentro de la plataforma MIFOS, en el encontraremos la siguiente información.

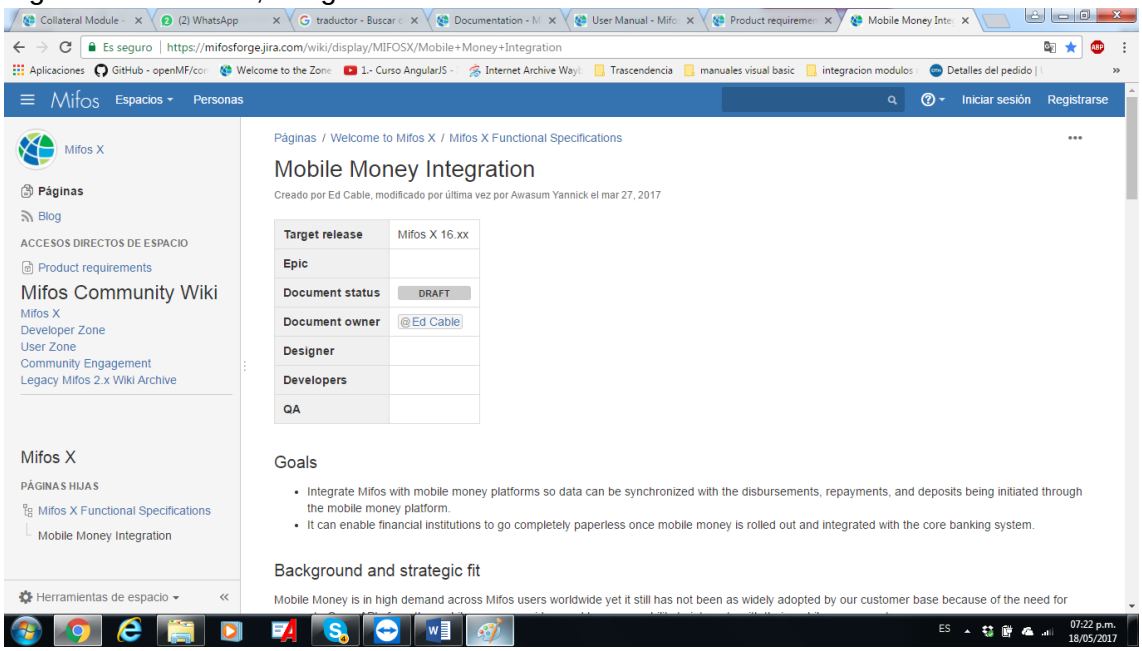
- Metas a las que quiere llegar el Usuario que propuso la mejora.
- Detalle del requerimiento sugerido por el Usuario.
- Diagramas de BD, Clase, Requerimientos funcionales.
- Documentos anexos código fuente, archivos planos, contactos.

Figura 39. Jira, ingreso de últimas actualizaciones de wiki Jira.



Fuente: wiki Jira, requerimientos funcionales de MIFOS.

Figura 40. Wiki Jira, Integración de modulo banca móvil.



Fuente: wiki Jira, requerimientos funcionales de MIFOS, localización de segundo requerimiento para modelar como decisión.

La necesidad fue propuesta por un miembro de la comunidad que se hace llamar por alias de **Awasum Yannick.**

Tabla 7. Datos de contacto, necesidad y solución planteada por el motivador de necesidad.

Datos Personales del motivador	Necesidad manifestada	Solución propuesta
Nombre completo: Awasum Yannick **Correo:** yannickawasum@gmail.com **País de origen:** Cameron. **Teléfono:** +237 681922919	Módulo de garantías para evitar replicación de información, cada vez que se generaba un crédito.	Mejorar el módulo de garantías, para evitar replicación de información, cuando se ingresa la información a crediticia del cliente.

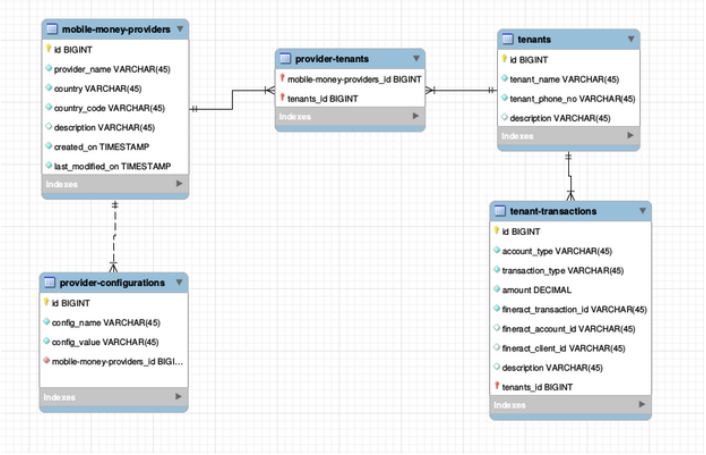
Fuente: https://mifosforge.jira.com/wiki/display/MIFOSX/Mobile+Money+Integration.

4.2.4.1 Lo manifestado Awasum Yannick. "La Banca móvil está en alta demanda a través de los usuarios de MIFOS en todo el mundo, pero todavía no ha sido adoptada por nuestra base de clientes debido a la necesidad de acceso a APIs de código abierto que deen ser proporcionadas por los proveedores de dinero móvil e implementar una integración de estas APIs a la arquitectura que se utiliza en MIFOS.

4.2.4.2 Esto es lo que propone Awasum Yannick. La Integración desde el punto de vista técnico no es muy compleja, una vez que el proveedor de servicios de banca móvil nos provea las APIs necesarias para integrar MIFOS con sus sistemas, también es necesario un apoyo de la comunidad MIFOS para que ayuden a construir los requerimientos funcionales, debido a que son ellos los más interesados en que esta nueva funcionalidad sea integrada de la manera más rápida, eficaz y funcionalmente.

En este modelo se implementó un modelo Entidad Relación para anexar las diferentes entidades de base de datos necesarias para integrar el módulo de Banca móvil, a su vez se diseñaron los casos de uso necesarios para complementar el modelo de base de datos y se citan algunos proveedores de APIS para Banca Móvil.

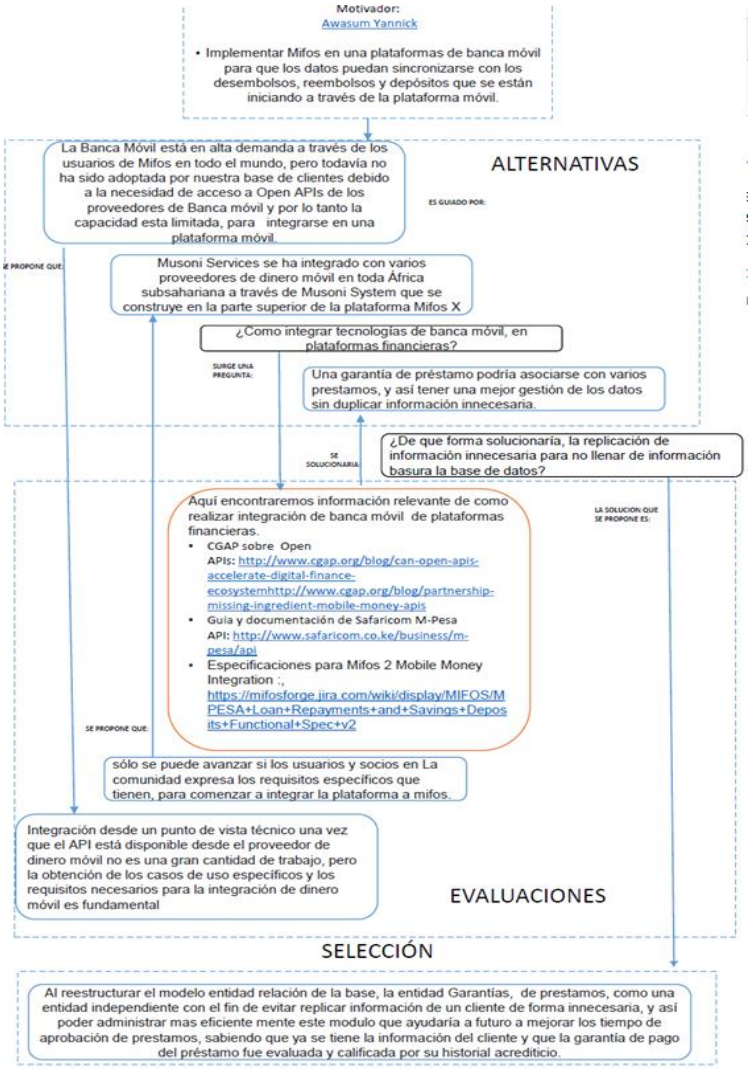
Figura 41. Modelo entidad Relación para implementación de módulo Bancarización Móvil.



Fuente: wiki Jira, requerimientos funcionales MIFOS

4.2.4.2 Modelo de decisiones N° 2 Motivador "Awasum Yannick"

Figura 42. Segundo modelo de decisiones, implementación de módulo Banca Móvil.



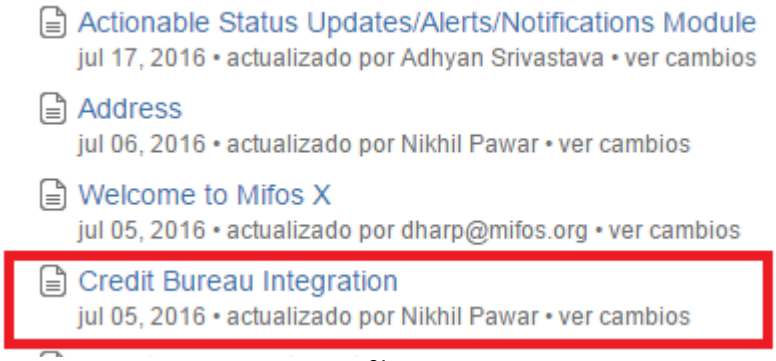
Fuente: Autores.

4.2.5 Modelo de anotaciones, integración de oficina de créditos MIFOS X. En este modelo, solo se ha llegado a la fase de documentación, se anexo un documento, en cual se indica dónde y cómo quedarían diseñados las nuevas integraciones de ingreso de sucursales de crédito desde cualquier parte del planeta y para cualquier modelo financiero existente en el planeta.

4.2.5.1 Módulo de integración de banca móvil. Este es nuestro objeto de estudio, un requerimiento plateado por un usuario el cual manifiesta la necesidad de implementar este módulo dentro de la plataforma MIFOS, en el encontraremos la siguiente información:

- Metas a las que quiere llegar el Usuario que propuso la mejora.
- Detalle del requerimiento sugerido por el Usuario.
- Diagramas de BD, Clase, Requerimientos funcionales.
- Documentos anexos código fuente, archivos planos, contactos.

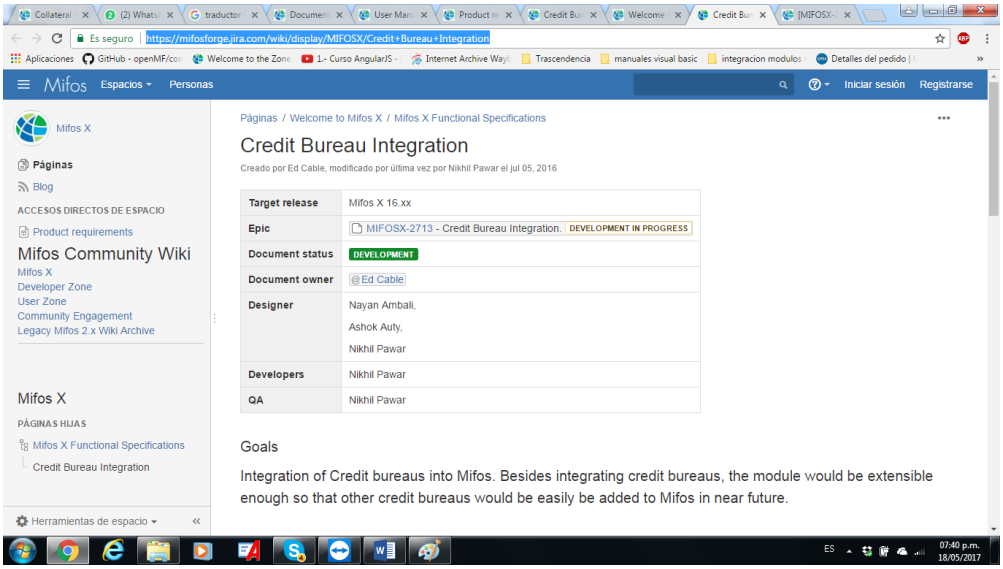
Figura 43. Wiki Jira, ingreso de últimas actualizaciones de wiki Jira.



Fuente: Jira, requerimientos MIFOS [21].

Figura 44. Wiki Jira, Modulo de integración de oficinas de crédito.

[21] Awasum Yannick Wiki jira, Modulo de banca móvil. [enero 2015]. <https://mifosforge.jira.com/wiki/display/MIFOSX/Mobile+Money+Integration>. [Citado 12 marzo de 2017]



Fuente: wiki Jira, requerimientos funcionales.

En el siguiente modelo de anotaciones, participaron más de un motivador, debido a la naturaleza del requerimiento, el módulo llamado “Integración de oficinas de crédito”, es una función de MIFOS que ya está implementada dentro de la aplicación base “la que ofrece la compañía”, pero según estos usuarios este módulo es demasiado básico para cubrir las necesidades requeridas en sus países de origen.

La necesidad fue propuesta por varios miembros de la comunidad que se hace llamar por alias de **Awasum Yannick, Nayan Ambali, Ashok Auty, Nikhil Pawar:**

Tabla 8. Datos de contacto, necesidad y solución planteada por el motivador de necesidad.

Datos Personales del motivador	Necesidad manifestada	Solución propuesta
Nombre completo: Nikhil Pawar **Correo:** yannickawasum@gmail.com **País de origen:** Cameron. **Teléfono:** +237 681922919	Al integrar esta funcionalidad, al módulo de sucursales de Crédito, el usuario tendrá la posibilidad de ingresar una sucursal, no importa el modelo financiero, ni tampoco los productos ya que va a tener la capacidad de hacerlo, ingresando tanto los tipos de productos, características, condiciones, plazos y	Modelo completo con el mockups, pertinentes para cada formulario y relación entre cada uno.

	demás.	
Nombre completo: Nayan Ambali **Correo:** nikhilpawar@yahoo.in **País de origen:** EEUU.	Al integrar esta funcionalidad, al módulo de sucursales de Crédito, el usuario tendrá la posibilidad de ingresar una sucursal, no importa el modelo financiero, ni tampoco los productos ya que va a tener la capacidad de hacerlo, ingresando tanto los tipos de productos, características, condiciones, plazos y demás.	Modelo completo con el mockups, pertinentes para cada formulario y relación entre cada uno.

Fuente: Jira Wiki[22]

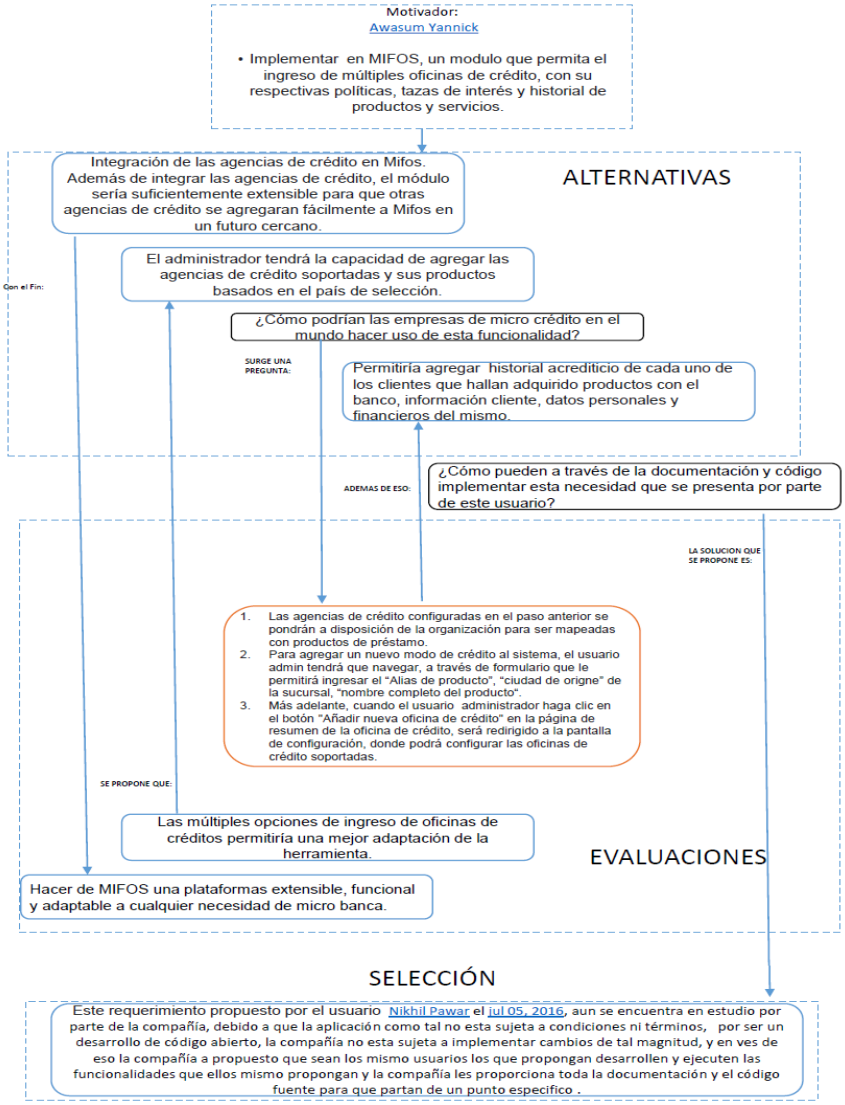
4.2.5.2 Lo manifestado Nikhil Pawar. Las sucursales de crédito, tienen diferentes requisitos, productos y leyes que los rigen, según su país de origen, MIFOS está muy ligado al modelo EUROPEO, el cual cada una de sus sucursales trabajan bajo las leyes de **unión europea** y sus distintas leyes gubernamentales, dejando así a los demás continentes limitados en cuanto al ingreso de sus productos

4.2.5.3 Esto es lo que propone Nikhil Pawar. Al integrar un nuevo módulo de ingreso de sucursales de crédito con un modelo más general, este podría adaptarse a cualquier necesidad de negocio en cualquier parte del mundo dejando así el paradigma gubernamental y las leyes que los rigen, se desarrolló un documento en el que muestra un anteproyecto, incluyendo mockups, requerimientos funcionales entre otra documentación.

4.2.5.4 Modelo de decisiones N° 3 Motivadores "Nayan Ambali, Ashok Auty,Nikhil Pawar"

Figura 45. Tercer modelo de anotaciones, Integración de oficinas de crédito.

[22]Awasum Yannick, Wiki jira. Óp., cit

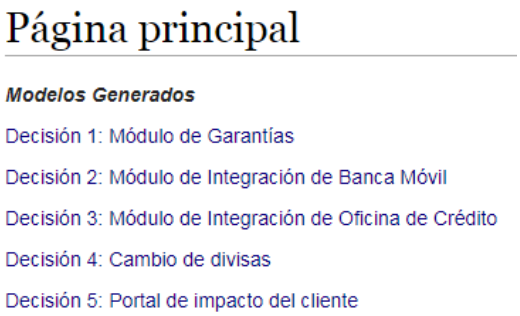


Fuente: Autores.

4.7 Representación de resultados en la Wiki Semántica

La primera parte de la wiki corresponde a decisiones que se elaboraron en el transcurso del proyecto.

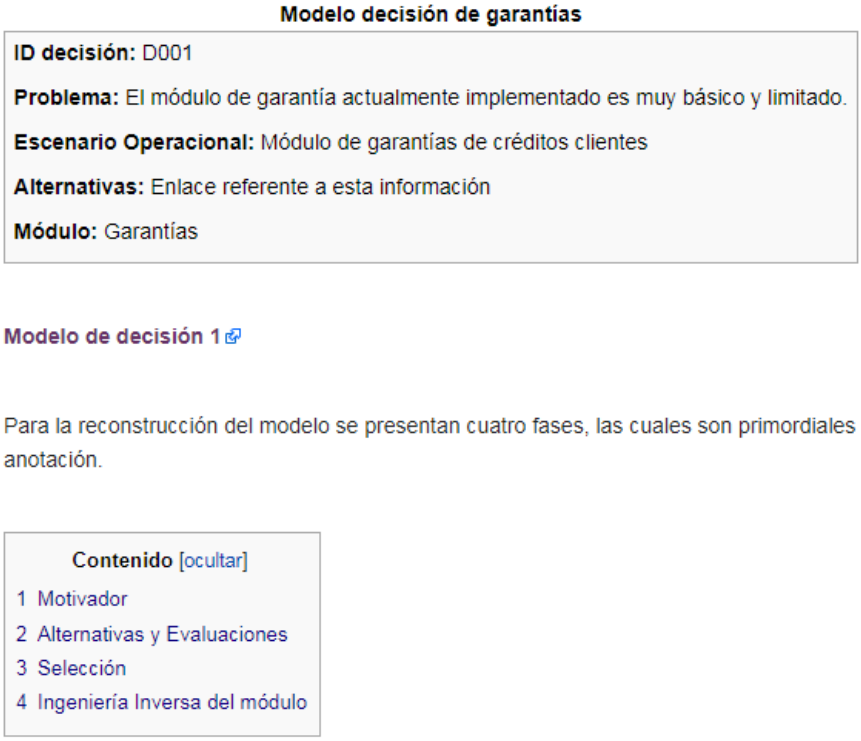
Figura 46. Decisiones de diseño reconstruidas de la herramienta MIFOS X.



Fuente: Autores.

A continuación, se muestra la estructura general que presenta cada una de las decisiones tomando de esta manera la información más relevante respecto a cada decisión.

Figura 47. Estructura de contenido de la primera decisión reconstruida.



Modelo decisión de garantías

ID decisión: D001

Problema: El módulo de garantía actualmente implementado es muy básico y limitado.

Escenario Operacional: Módulo de garantías de créditos clientes

Alternativas: Enlace referente a esta información

Módulo: Garantías

Modelo de decisión 1

Para la reconstrucción del modelo se presentan cuatro fases, las cuales son primordiales al momento de recuperar la anotación.

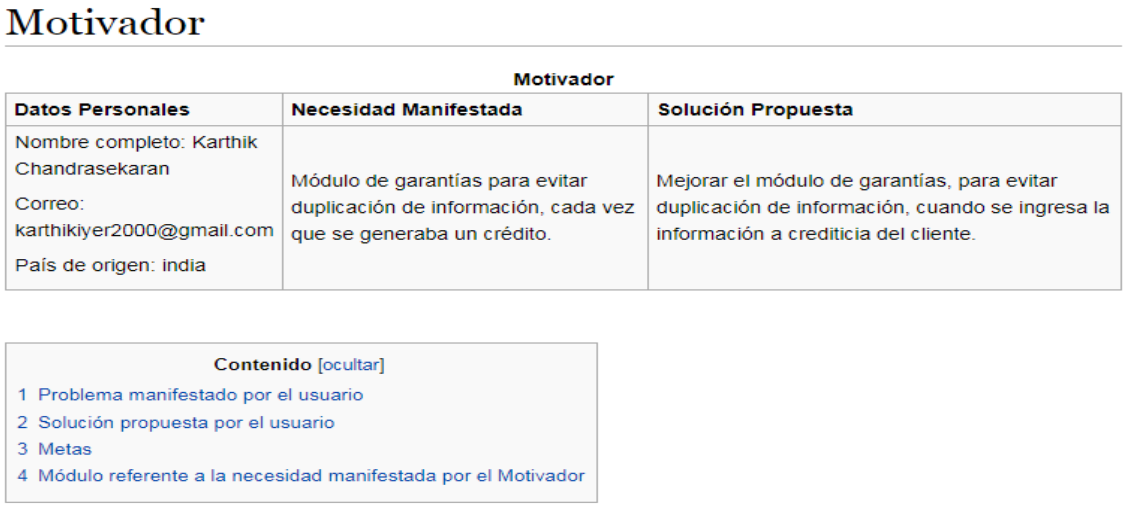
Contenido [ocultar]

1 Motivador
2 Alternativas y Evaluaciones
3 Selección
4 Ingeniería Inversa del módulo

Fuente: Autores

En la información referente al motivador se encuentra todos los datos referentes a la persona o personas que iniciaron dicha motivación por medio de una necesidad.

Figura 48. Información del motivador, primera decisión reconstruida



Motivador

Motivador

Datos Personales	Necesidad Manifestada	Solución Propuesta
Nombre completo: Karthik Chandrasekaran Correo: karthikiyer2000@gmail.com País de origen: india	Módulo de garantías para evitar duplicación de información, cada vez que se generaba un crédito.	Mejorar el módulo de garantías, para evitar duplicación de información, cuando se ingresa la información a crediticia del cliente.

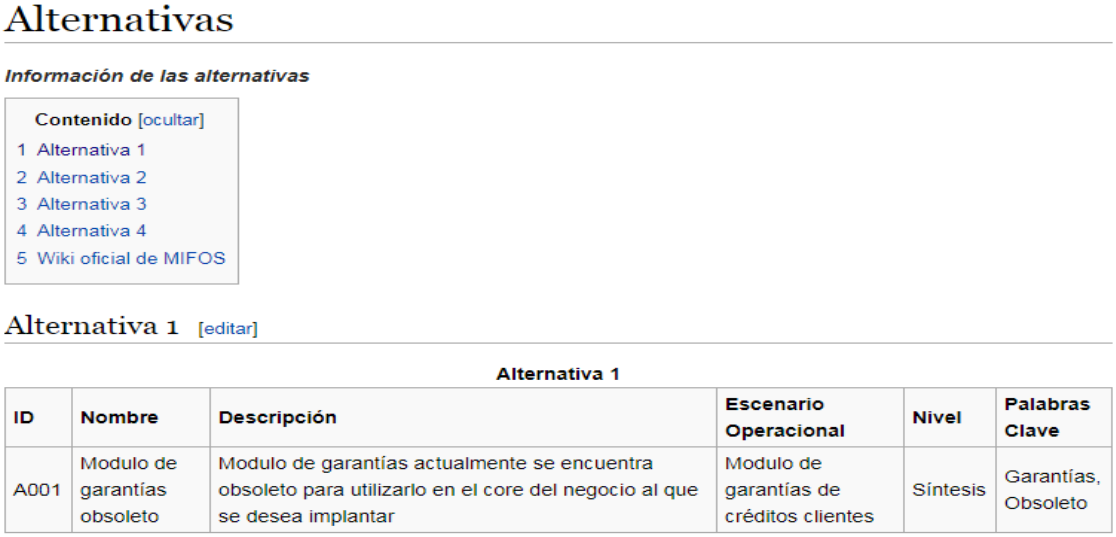
Contenido [ocultar]

1 Problema manifestado por el usuario
2 Solución propuesta por el usuario
3 Metas
4 Módulo referente a la necesidad manifestada por el Motivador

Fuente: Autores

En la información de las alternativas se informa de la cantidad de alternativas que componen la decisión dicho número puede variar según la decisión, así mismo se muestra las respectivas evaluaciones que componen a cada una de las alternativas.

Figura 49. Información de las alternativas y evaluaciones que componen la decisión.



Alternativas

Información de las alternativas

Contenido [ocultar]
1 Alternativa 1
2 Alternativa 2
3 Alternativa 3
4 Alternativa 4
5 Wiki oficial de MIFOS

Alternativa 1 [editar]

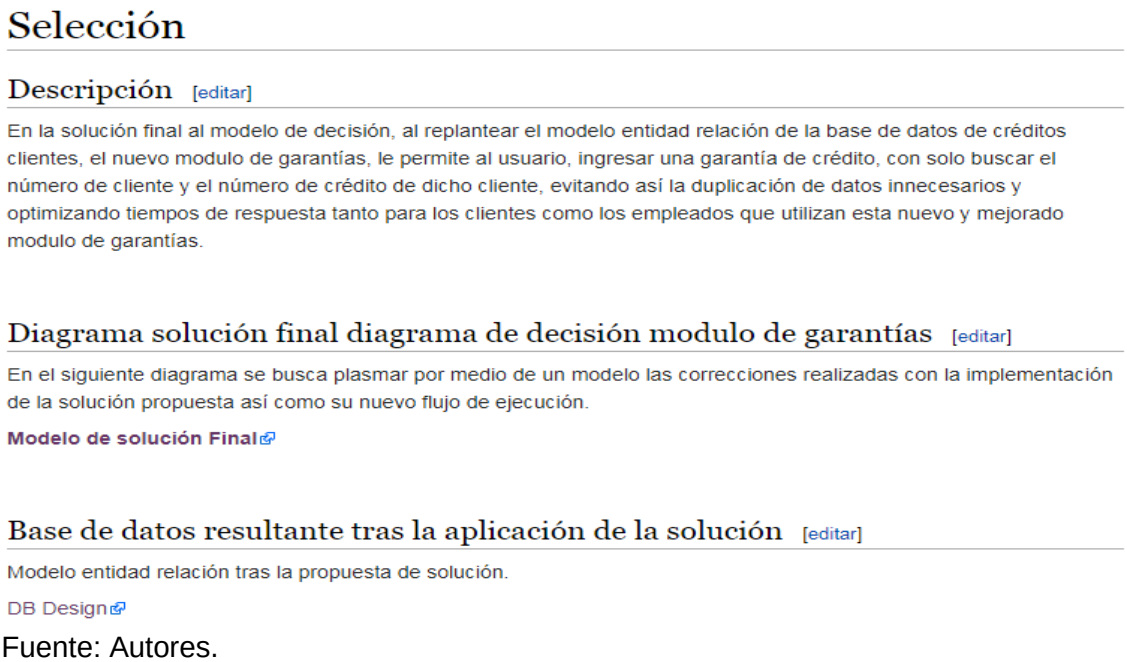
Alternativa 1

ID	Nombre	Descripción	Escenario Operacional	Nivel	Palabras Clave
A001	Modulo de garantías obsoleto	Modulo de garantías actualmente se encuentra obsoleto para utilizarlo en el core del negocio al que se desea implantar	Modulo de garantías de créditos clientes	Síntesis	Garantías, Obsoleto

Fuente: wiki jira, requerimientos funcionales MIFOS, módulo de garantías.

En la solución final se muestra la información relevante a la solución obtenida tras la reconstrucción de la decisión, así como las necesidades que esta logró cubrir.

Figura 50.Solución final de la primera decisión.



Selección

Descripción [editar]

En la solución final al modelo de decisión, al replantear el modelo entidad relación de la base de datos de créditos clientes, el nuevo modulo de garantías, le permite al usuario, ingresar una garantía de crédito, con solo buscar el número de cliente y el número de crédito de dicho cliente, evitando así la duplicación de datos innecesarios y optimizando tiempos de respuesta tanto para los clientes como los empleados que utilizan esta nuevo y mejorado modulo de garantías.

Diagrama solución final diagrama de decisión modulo de garantías [editar]

En el siguiente diagrama se busca plasmar por medio de un modelo las correcciones realizadas con la implementación de la solución propuesta así como su nuevo flujo de ejecución.

Modelo de solución Final

Base de datos resultante tras la aplicación de la solución [editar]

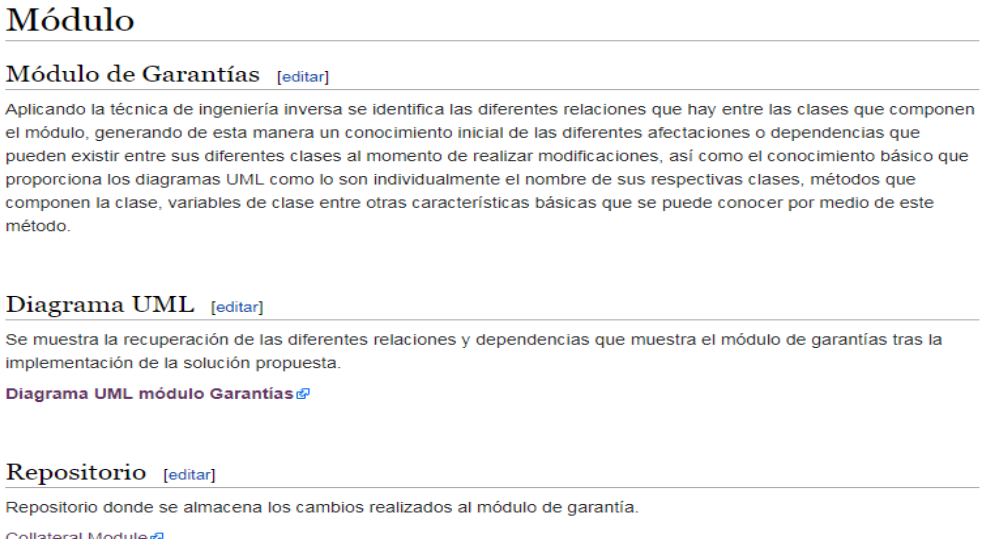
Modelo entidad relación tras la propuesta de solución.

DB Design

Fuente: Autores.

Finalmente se muestra el proceso de ingeniería inversa aplicada al respectivo módulo en cual fue modificado en la presente decisión.

Figura 51. Ingeniería inversa aplicada al módulo referente de la primera decisión.



Módulo

Módulo de Garantías [editar]

Aplicando la técnica de ingeniería inversa se identifica las diferentes relaciones que hay entre las clases que componen el módulo, generando de esta manera un conocimiento inicial de las diferentes afectaciones o dependencias que pueden existir entre sus diferentes clases al momento de realizar modificaciones, así como el conocimiento básico que proporciona los diagramas UML como lo son individualmente el nombre de sus respectivas clases, métodos que componen la clase, variables de clase entre otras características básicas que se puede conocer por medio de este método.

Diagrama UML [editar]

Se muestra la recuperación de las diferentes relaciones y dependencias que muestra el módulo de garantías tras la implementación de la solución propuesta.

Diagrama UML módulo Garantías

Repositorio [editar]

Repositorio donde se almacena los cambios realizados al módulo de garantía.

Collateral Module

Fuente: Autores.

5. CONCLUSIONES

Es importante mencionar que el actual proyecto corresponde a la fase primaria de un complejo tema que trata de recuperar decisiones de diseño como conocimiento en el contexto de una herramienta open source. Este conocimiento puede variar en tiempos para la completa recuperación esto depende de la documentación de la herramienta y del entendimiento del modelo de negocio que esta desea abarcar. La técnica propuesta consiste en documentar las decisiones después de cumplido el hecho, por lo que la técnica no es intrusiva y permite evolucionar y completar el conocimiento parcial.

Para la herramienta MIFOS X que busca cubrir la necesidad de un core bancario en entidades financieras es un tema bastante complejo pero que se debe entender bastante bien con la finalidad de que dado el caso de modificar esta herramienta a necesidad dicha modificación no se encuentre fuera de los parámetros establecidos de la lógica de negocio.

Por otra parte, se hace énfasis en que la principal técnica utilizada en el presente proyecto es el ciclo de vida del conocimiento, con la finalidad de proporcionar a nuevos usuarios información relevante como las decisiones tomadas para el desarrollo o mejora de algunos módulos propuesta por usuarios de la comunidad MIFOS, es decir, generar un conocimiento más teórico que técnico.

Mencionado lo anterior se indica que el conocimiento obtenido por el proceso de ingeniería inversa obtenido es solo la base para futuras investigaciones realizadas en la herramienta MIFOS X teniendo como tema principal proporcionar un conocimiento más técnico.

Tras la finalización del presente proyecto se puede concluir en primera instancia que es viable para usuarios que quieran incurrir en esta herramienta, la recuperación de decisiones ya que esto brinda tener una mayor comprensión de la arquitectura de la herramienta, así como la lógica de negocio que esta desea cubrir, por lo cual debido a la complejidad de la herramienta se vuelve una necesidad contar con este conocimiento recuperado.

Por otra parte, es importante mencionar que la recuperación de conocimiento es parcial. La ventaja es que a través de la wiki semántica se podrá llegar al 100% de la herramienta después de un tiempo significativo por lo cual se busca automatizar este factor con el fin de reducir tiempos en dicha recuperación.

Tras la experiencia adquirida durante el desarrollo del presente proyecto se puede afirmar que la administración del conocimiento, así como sus diferentes técnicas es completamente aplicable a la herramienta MIFOS X, ya que cumple con todos los requisitos para la aplicación de dichos recursos.

Finalmente se afirma que tras la finalización del proceso de reconstrucción de decisiones los resultados obtenidos generan una mayor coherencia en cuanto a la estructuración del conocimiento recuperado y el cual es más fácil de entender para usuarios que están iniciando en el uso de la herramienta MIFOS X.

BIBLIOGRAFÍA

BANKING OFFERING. accenture consulting. [Online]. United Kingdom. [Fecha de consulta: febrero 12 de 2017]. Disponible en: https://www.accenture.com/gb-en/core-banking-services.

COMUNIDAD MIFOS. Mifos Initiative. [Online]. United State. [Fecha de consulta: agosto 2 de 2016]. Disponible en: http://mifos.org/.

GONZÁLES, Oscar. PEDRAZA-GARCÍA, Gilberto. CORREAL, Dario. BELTRAN, Guillermo. “MONO+KM: Knowledge Management in Collaborative Project Development,” in Ingenieria y Universidad, [S.l.], v. 20, n. 2, jun. 2016. ISSN 2011-2769.

HERRAMIENTA DE INGENIERÍA INVERSA. ObjectAid UML. [Online]. Estados Unidos. [Fecha de consulta: febrero 20 de 2017], Disponible en: http://www.objectaid.com/.

HISTORIAL DE VERSIONES. Archivo páginas web. [Online]. Estados Unidos. [Fecha de consulta: febrero 2 de 2017], Disponible en: https://archive.org/.

IKUJIRO. Nonaka, HIROTAKA. Takeuchi (1995). “The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation,” in Oxford University Press. ISBN: 0195092694.

KIMIZ. Dalkir. “Knowledge Management in Theory and Practice,” in The MIT Press. 2011. ISBN: 0262015080, 9780262015080.

MIFOS X. enciclopedia libre Wikipedia. [Online]. Estados Unidos. [Fecha de consulta: agosto 14 de 2016]. Disponible en: https://en.wikipedia.org/wiki/Mifos_X.

NOVELO. Gustavo. Conocimiento Tácito y Explícito. [Online]. México. Knowledge Management. [Fecha de consulta: marzo 17 de 2017]. Disponible en: https://www.youtube.com/watch?v=4pq7SsvAHl4

PEDRAZA-GARCÍA, Gilberto. ASTUDILLO, Hernán. CORREAL, Dario. “DVIA: Understanding how software architects make decisions in design meetings,” in Proceedings of the 2015 European Conference on Software Architecture Workshops (ECSAW '15). ACM, New York, NY, USA, Article 51, 2015, 7 pages. DOI=10.1145/2797433.2797486 http://doi.acm.org/10.1145/2797433.2797486 http://dl.acm.org/citation.cfm?doid=2797433.2797486.

PLATAFORMA MIFOS X. Mifos X. [Online]. Estados Unidos. [Fecha de consulta: agosto 1 de 2016]. Disponible en: http://mifos.org/mifos-x/

RAMOS-ACOSTA. Diego Alonso, Proceso de ingeniería inversa. [Online]. Colombia. Escuela Colombiana de Ingeniería. [Fecha de consulta: marzo 23 de 2017]. Disponible en: https://www.acofipapers.org/index.php/acofipapers/2013/paper/viewFile/380/189.

REPOSITORIO BACK. Back end plataforma MIFOS X. [Online]. Estados Unidos. [Fecha de consulta: septiembre 1 de 2016]. Disponible en: https://github.com/openMF/mifosx.

REPOSITORIO FRONT. from end plataforma MIFOS X. [Online]. Estados Unidos. [Fecha de consulta: septiembre 1 de 2016] Disponible en: https://github.com/openMF/community-app#mifosx-community-app.

REQUERIMIENTO ESPECIFICO. Módulo de garantías MIFOS X. [Online]. Estados Unidos. [Fecha de consulta: marzo 20 de 2017] Disponible en: https://mifosforge.jira.com/wiki/spaces/MIFOSX/pages/149749763/Collateral+Module.

WIKI COMUNIDAD MIFOS. Mifos Jira. [Online]. Estados Unidos. [Fecha de consulta: febrero 15 de 2017]. Disponible en: https://mifosforge.jira.com/secure/Dashboard.jspa.

ANEXOS

Anexo A. Instalación y uso de ObjectAid

Para la instalación de la herramienta ObjectAid se debe contar previamente con el IDE de desarrollo Eclipse, una vez en Eclipse se deberá seguir las siguientes instrucciones:

1. En el menú principal de Eclipse, vaya a Ayuda> Instalar nuevo software...

2. En la página 'Software disponible' del asistente 'Instalar', pulse el botón 'Añadir ...'

3. En el cuadro de diálogo 'Añadir repositorio', ingrese esta información y pulse 'Aceptar':
Nombre: ObjectAid UML Explorer
URL: http://www.objectaid.com/update

4. El cuadro de diálogo 'Instalar' mostrará ahora los plug-ins de ObjectAid disponibles. Seleccione lo que desea instalar y pulse 'Siguiente'. Puede que desee desactivar la casilla de verificación "Contactar todos los sitios de actualización durante la instalación para encontrar el software necesario" para acelerar la descarga.

5. En la página "Detalles de la instalación" debería ser posible simplemente pulsar 'Siguiente'.

6. En la página 'Revisión de licencias' debe aceptar la licencia de ObjectAid y pulsar 'Finalizar' para comenzar la instalación.

7. Recibirá un mensaje de "Advertencia de seguridad" porque los JAR de ObjectAid no están firmados. Presione 'OK' para continuar.

8. Una vez finalizada la instalación, se le preguntará si desea "Reiniciar ahora", "Ahora no" o "Aplicar cambios ahora". Para estar seguro, presione el botón 'Reiniciar ahora' y el Explorador UML de ObjectAid estará disponible después del reinicio.

Una vez instalada la herramienta el uso de la herramienta consta de seguir una serie de pasos para obtener resultados satisfactorios:

Primero se crea un diagrama de clase vacío con el asistente 'Nuevo'. Para llegar allí, simplemente puede presionar Ctrl + N en el paquete o carpeta donde desea crear su diagrama de clases.

A continuación, arrastra algunas clases en su diagrama, normalmente desde el Explorador de paquetes. Cualquier otra vista que contenga tipos Java funcionará también, p. La jerarquía de tipos, la jerarquía de llamadas y las vistas de búsqueda.

Al arrastrar una clase dentro del diagrama de clases este genera automáticamente un diagrama UML con toda la información referente a la clase seleccionada.

Luego de tener la cantidad de clases deseadas se debe conectar cada una dando click derecho y dando la opción de generar todas las relaciones, siguiendo este proceso con cada una de las clases se obtendrá las diferentes relaciones entre las diferentes clases según la lógica del proyecto que contiene dichas clases.